Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Diego Brito de Carvalho

Dois Códigos Corretores de Erro BCH para Comunicações Ópticas: Implementação em FPGA e Comparações

Campinas
2015

Diego Brito de Carvalho

Dois Códigos Corretores de Erro BCH para Comunicações Ópticas: Implementação em FPGA e Comparações

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Telecomunicações e Telemática.

Orientador: Max Henrique Machado Costa

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO DIEGO BRITO DE CARVALHO, E ORIENTADA PELO PROF. DR. MAX HENRIQUE MACHADO COSTA.

_____

Campinas
2015

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

C253d

Carvalho, Diego Brito de, 1984-
Dois códigos corretores de erro BCH para comunicações ópticas : implementação em FPGA e comparações / Diego Brito de Carvalho. – Campinas, SP : [s.n.], 2015.

Orientador: Max Henrique Machado Costa.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Códigos corretores de erros (Teoria da informação). 2. Lógica programável - Dispositivos. 3. Comunicações ópticas. I. Costa, Max Henrique Machado,1950-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Two BCH error correcting codes for optical communications : implementation in FPGA and comparisons
Palavras-chave em inglês:
Code error correcting (Information theory)
Programmable Logic - Devices
Optical communications
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Max Henrique Machado Costa [Orientador]
Cecilio José Lins Pimentel
Darli Augusto de Arruda Mello
Data de defesa: 14-12-2015
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Diego Brito de Carvalho RA: 085474
Data da Defesa: 14/12/2015
Título da Tese: "Dois Códigos BCH para Comunicações Ópticas: Implementação em FPGA e Comparações"

Prof. Dr. Max Henrique Machado Costa (Presidente, FEEC/UNICAMP)
Prof. Dr. Cecilio José Lins Pimentel (UFPE)
Prof. Dr. Darli Augusto de Arruda Mello (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Dedico este trabalho à memória de meu pai, Rubens Ferreira Carvalho, por guiar meus passos na escolha de minha carreira, e à minha mãe, Sônia Nascimento Brito de Carvalho pelo incentivo e apoio incondicionais.

Agradecimentos

Agradeço primeiramente a Deus a saúde, a coragem e a perseverança durante esta caminhada.

Agradeço aos meus pais, Rubens Ferreira Carvalho e Sônia Nascimento Brito de Carvalho, que sempre estiveram presentes em minha vida, proporcionando minha formação pessoal e profissional. Obrigado pai, por me ensinar valores que moldaram o meu caráter. Obrigado mãe, pelo amor, zelo e dedicação incansáveis. Eu não seria absolutamente nada sem o que vocês fizeram por mim.

Agradeço à minha esposa, Mariana Carvalho de Almeida, o companheirismo nas alegrias e nas tristezas, o apoio e a paciência comigo durante o tempo dedicado à realização deste trabalho.

Agradeço ao Prof. Dr. Max Henrique Machado Costa a valiosa orientação e também os ensinamentos dentro e fora da sala de aula.

Agradeço à Fundação CPqD (Centro de Pesquisa e desenvolvimento em Telecomunicações). Obrigado por me acolher desde o início de minha carreira profissional, por permitir que eu me ausentasse para assistir às aulas de pós-graduação e também por ceder a infraestrutura necessária para o desenvolvimento deste trabalho.

Agradeço à Padtec S/A e à Fundação CPqD a permissão para publicar este trabalho.

Agradeço a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho. Aos amigos e colegas de trabalho Antônio Unias de Lucena, Érico Nunes Ferreira Bastos, Giuliano Ferronato, Luiz Juberto Rossi de Jesus, Marcelo Marcos Polidoro e Rodolfo Soares Caproni agradeço o esforço e empenho dedicados à realização do projeto que deu origem a este trabalho. Ao Arley Henrique Salvador agradeço a parceria na produção de conteúdo científico. Agradeço especialmente ao Juberto e ao Polidoro as tantas vezes em que compartilharam seus conhecimentos comigo, contribuindo significativamente para o meu aprimoramento profissional.

Dedico o agradecimento final à Unicamp, que me aceitou em seu corpo discente e contribuiu para a minha formação.

O dinheiro faz homens ricos, o conhecimento faz homens sábios e a humildade faz grandes homens.

Mahatma Gandhi

Resumo

Atualmente as redes de transporte óptico têm papel fundamental nas telecomunicações mundiais graças às características das fibras ópticas, tais como baixa perda e alta banda passante. Com a evolução desse tipo de tecnologia, grandes quantidades de dados são transmitidos ao redor do globo terrestre através de enlaces de dimensões intercontinentais.
O recente aumento do tráfego de informação nas redes de telecomunicações, causado principalmente por fatores como a popularização dos dispositivos móveis inteligentes e também pela transmissão de vídeo, gera a necessidade de se pesquisar alternativas que concedam ainda mais capacidade e robustez às redes ópticas existentes. Nesse sentido, este trabalho apresenta dados e comparações entre duas implementações em FPGA de um código corretor de erro para comunicações ópticas com alto ganho de codificação, que utiliza dois códigos BCH(1020,988) entrelaçados: a primeira segue estritamente o esquema proposto no anexo I.9 da recomendação do ITU-T G.975.1, enquanto a segunda, derivada da primeira, se vale de um aumento da redundância (paridade) e da reorganização das estruturas de quadro para produzir um ganho de codificação superior ao do código original.

Palavras-chave: códigos corretores de erro, códigos de bloco, codificação de canal, implementação em lógica programável, comunicações ópticas, redes de transporte óptico.

Abstract

Currently, optical transport networks play a fundamental role in global telecommunications thanks to the characteristics of optical fibers, such as low loss and high bandwidth. With the evolution of this type of technology, large amounts of data can be transmitted around the globe through links of intercontinental dimensions.
The recent increase in data traffic on telecommunication networks, mainly caused by factors such as the popularity of smart mobile devices and also by video transmission, generates the need to investigate alternatives that provide even more capacity and robustness to existing optical networks. In this sense, this dissertation presents data and comparisons between two implementations in FPGA of an error correction code for optical communications with high coding gain, which uses two interleaved BCH(1020.988) codes: the first one strictly follows the arrangements proposed in annex I. 9 of ITU-T recommendation G.975.1, while the second one, deriving from the first, relies on increasing redundancy (parity) and reorganizing the frame structures to produce a coding gain superior to that of the original code.

Key words: error correcting codes, block codes, channel coding, implementation in programmable logic, optical communications, optical transport networks.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos e Notação

ASIC	*Application-Specific Integrated Circuit*
AWGN	*Additive White Gaussian Noise*
BCH	Bose-Chaudhuri-Hocquenqhem
BDI	*Backward Error Indication*
BEI	*Backward Error Indication*
BER	*Bit Error Rate*
BIP-8	*Bit Interleaved Parity-8*
CBR	*Constant Bit Rate*
CD	*Compact Disc*
CDMA	*Code Division Multiple Access*
CPqD	Centro de Pesquisa e Desenvolvimento em Telecomunicações
CSF	*Client Signal Fail*
DCF	*Dispersion Compensating Fiber*
DWDM	*Dense Wavelength Division Multiplexing*
EFEC	*Enhanced Forward Error Correction*
erfc	*Complementary Error Function*
FAS	*Frame Alignment Signal*
FBG	*Fiber Bragg Grating*
FEC	*Forward Error Correction*
FIFO	*First In First Out*
FPGA	*Field Programmable Gate Array*
GCC0	*General Communications Channel*
GF	Galois *Field*
HD	*Hard-Disk*
HF	*High Frequency*
IIS	Interferência Intersimbólica
IP	*Internet Protocol*
ISI	*Intersymbol Interference*
ITU	*International Telecommunication Union*
LDPC	*Low-Density Parity Check*
LIT	Linear e Invariante no Tempo
MFAS	*Multiframe Alignment Signal*
ML	*Maximum Likelihood*

NCG	*Net Coding Gain*
ODU	*Optical Data Unit*
OH	*Overhead*
OPU	*Optical Payload Unit*
OTN	*Optical Transport Network*
PDH	*Plesyochronous Digital Hierarchy*
P2P	*Peer-to-Peer*
PSI	*Payload Structure Identifier*
PSTN	*Public Switched Telefone Network*
RAM	*Random-Access Memory*
ROM	*Read-Only Memory*
RS	Reed-Solomon
SDH	*Synchronous Digital Hierarchy*
SG15	Grupo de Estudo 15 do ITU-T
SM	*Section Monitoring*
SNR	*Signal-to-Noise Ratio*
TDM	*Time Division Multiplexing*
TTI	*Trail Trace Identifier*
SONET	*Synchronous Optical Network*
VHDL	*VHSIC Hardware Description Language*
WDM	*Wavelenght Division Multiplexing*

$\forall$	Notação que significa para qualquer.
$\wedge$	Conectivo lógico que significa a operação lógica "E".
$\in$	Notação que significa pertence ao conjunto.
$\mid$	Notação que significa tal que.
$(G, *)$	Conjunto de elementos com a operação $*$ definida entre eles.
$(A, +, \cdot)$	Conjunto de elementos com as operações $+$ e $\cdot$ definidas entre eles.
$C(n, k)$	Notação para representar os parâmetros de um código corretor de erros.
$\mathbb{Z}$	Conjunto dos números inteiros. $\mathbb{Z} = \{... - 2, -1, 0, 1, 2...\}$.

Sumário

Prólogo

A tecnologia pode ser caracterizada por tudo aquilo que o homem cria visando aprimorar e facilitar a sua interação com o planeta. Desde a Pré-História, quando o homem descobriu o fogo, criava simples instrumentos de pedra e praticava a agricultura rudimentar, ele vem protagonizando uma evolução tecnológica que passa por grandes momentos, tais como os das seguintes invenções e descobertas: o arado; novos materiais como o cobre, o bronze e o ferro fundido; a pólvora; o papel; os moinhos; os teares; a bússola; as caravelas, a imprensa; etc. Porém, após a Revolução Industrial, o ritmo da evolução tecnológica se acelera com as invenções desse período, dentre as quais se destacam a máquina a vapor e os motores a combustão, que propiciaram o desenvolvimento da indústria e dos transportes. Também foi durante esse período, mais especificamente durante a chamada Segunda Revolução Industrial, que se deu a ampliação do uso da eletricidade. Embora sua história tenha origem no século VI a.C., foi apenas a partir da segunda metade do século XIX que essa forma de energia começou a ser largamente utilizada. Após a invenção da lâmpada elétrica, dos motores elétricos, da válvula, dos transistores, dos semicondutores, dos circuitos integrados, dos computadores modernos e do laser, o homem vem vivenciando uma evolução tecnológica imensamente rápida nos setores energético, médico e principalmente nas comunicações.

A cada dia a tecnologia se faz mais presente na vida do homem. Esse fato se deve, em grande parte, à informatização da sociedade e à ampla disseminação do uso de aparelhos eletrônicos digitais no cotidiano das pessoas. É cada vez maior a variedade de dispositivos disponíveis no mercado que são capazes de tratar dados, seja armazenando-os apenas ou transmitindo-os de um lado ao outro do globo terrestre. Consequentemente, há um aumento na demanda por informação, o que leva à necessidade do constante aprimoramento das tecnologias de armazenamento e transmissão de dados a fim de expandir suas capacidades.

O presente trabalho apresenta uma arquitetura para um código corretor de erros de alto ganho que é empregado em transmissões ultrarrápidas por fibra óptica. Apresenta também uma arquitetura para um código derivado do citado anteriormente, mas que emprega maior redundância de informação a fim de se obter um aumento no valor do ganho em relação ao original.

Capítulo 1

Introdução

1.1 Visão Geral sobre Teoria de Informação

O armazenamento e a transmissão de dados são procedimentos comuns em sistemas digitais, nos quais a informação é submetida a um meio físico chamado de canal. Um circuito integrado, um HD (*hard-disk*), um cabo coaxial, o ar, a fibra óptica, etc., são exemplos de canais e todos eles estão sujeitos à ação de imperfeições que podem levar à introdução de erros na informação, tais como ruídos, interferências, distorções, desvanecimentos, etc. Dependendo do caso, a ocorrência de erros pode ser um grande problema pois algumas aplicações são muito pouco tolerantes. O ideal seria que os dados originais permanecessem íntegros durante o processo mas, como nem sempre isso é possível, torna-se necessário o estudo e o desenvolvimento de técnicas que contornem ou minimizem esses problemas.

No ano de 1948 o matemático, engenheiro eletrônico e criptógrafo americano Claude E. Shannon publicou o importante artigo científico *A Mathematical Theory of Communication*. Esse artigo é considerado como o marco que estabeleceu o ramo da teoria da probabilidade e estatística que trata temas como sistemas de comunicação digital, detecção e correção de erros e criptografia: a **teoria de informação**.

A teoria de informação também tem como pilares três teoremas demonstrados por Shannon: o teorema da codificação de fonte, conhecido como primeiro teorema de Shannon; o teorema da codificação de canal, conhecido como segundo teorema de Shannon; e o teorema da capacidade de canal, que é provavelmente o mais famoso e é conhecido como terceiro teorema de Shannon.

O teorema da codificação de canal permite mostrar que é possível submeter uma informação através de um canal ruidoso e recuperá-la com probabilidade de erros tão baixa

quanto desejado, desde que a taxa de transmissão seja menor do que a capacidade do canal. É principalmente sobre esse teorema que se apóiam as técnicas de correção de erro à frente (FEC, *forward error correction*) que são, basicamente, códigos que adicionam redundância calculada ao dado que será armazenado e/ou transmitido de modo que seja possível detectar e corrigir erros.

Os códigos corretores de erro vêm sendo usados com sucesso há bastante tempo. Em 1972, a nave espacial Mariner 9 transmitiu imagens de Marte utilizando o código Reed-Muller. As naves Voyager 1 e Voyager 2 transmitiram imagens de Júpiter e Saturno e seus satélites entre 1979 e 1980 utilizando um código Golay. A Voyager 2 posteriormente fotografou Urano e Netuno, mas teve seu código alterado para um código convolucional concatenado a um código de Reed-Solomon por este possuir maior capacidade de correção de erros.

Atualmente, o emprego de códigos corretores de erro é muito comum. Sempre que alguém faz uso de alguma informação digitalizada, seja quando fala ao celular, assiste televisão digital, grava ou ouve um CD (*compact disc*) de música, armazena uma foto tirada com uma câmera digital em um cartão de memória ou navega pela internet, está se beneficiando do uso de um código corretor de erro. Os sistemas e redes de comunicação via satélite, cabo e fibra óptica também utilizam FEC.

1.2 Motivação

Durante as últimas décadas, principalmente após o surgimento e a popularização da internet e mais recentemente após a intensificação do uso de computadores portáteis, telefones inteligentes e *tablets*, o volume do tráfego de informações tem crescido em ritmo acelerado. Segundo um artigo divulgado recentemente pela Cisco [1], o tráfego IP (*internet protocol*) global aumentou mais de cinco vezes nos últimos cinco anos e, em 2014, 67% desse volume foi composto por vídeo, sem incluir as transferências arquivos do tipo ponto-a-ponto (P2P, *peer-to-peer*). Por essas e outras razões, faz-se necessário o uso de taxas cada vez mais elevadas em redes de transmissão, com alta eficiência e confiabilidade.

As redes de transporte óptico (OTN, *optical transport network*) têm como código corretor de erro padrão o Reed-Solomon RS(255,239). Tal código é definido no anexo A da recomendação G.709 do ITU-T (divisão de normalização em telecomunicações do *Inter-*

national Telecommunication Union) [2] e concede 5,6 dB de ganho de codificação líquido (NCG, *net coding gain*) a uma taxa de erro de saída (BER_{out}, *output bit error rate*) igual a 10^{-12}. A fim de atender à alta demanda de tráfego de dados no núcleo da rede e também para tornar possível o aumento da distância entre os elementos de rede, novos códigos corretores de erro mais poderosos vêm sendo desenvolvidos. Aprovada pelo Grupo de Estudo 15 (SG15) do ITU-T, a recomendação G.975.1 [3] descreve oito códigos selecionados por meio de um trabalho de engenharia conjunta, cuja aplicação principal é sistemas ópticos submarinos com tecnologia de multiplexação densa por comprimento de onda (DWDM, *dense wavelength division multiplexing*). Os referidos códigos concedem ganho de codificação líquido superior ao fornecido pelo Reed-Solomon RS(255,239) e, por essa razão, podem ser considerados aprimorados e chamados de *Super* FEC ou EFEC (*enhanced forward error correction*).

Outra grande motivação partiu da indústria. A maior fabricante de equipamentos para comunicações ópticas do Brasil, Padtec S/A, identificou a necessidade de ter em sua linha de produtos um código corretor de erro para aplicação em redes OTN de 2,5 Gbps, implementável em dispositivo lógico programável FPGA (*field-programmable gate array*) e que fosse capaz de prover um NCG próximo a 10 dB a uma BER_{out} de 10^{-15}. Esse fato levou à criação de um projeto na Fundação CPqD, iniciado em agosto de 2010 e finalizado em 2013, que deu origem a este trabalho. Com a consultoria da Zeuxion, uma DH (*design house*) dinamarquesa especializada em telecomunicações, a equipe do CPqD foi responsável principalmente pela concepção das arquiteturas finais de alto e baixo níveis dos codificadores e decodificadores abordados neste trabalho, pela implementação em FPGA dos mesmos, pela montagem do arranjo experimental e execução dos testes de laboratório e pela elaboração de toda a documentação envolvida na aprovação e aceite do projeto por parte do cliente. Inicialmente a Padtec submeteu um documento contendo os requisitos a serem atendidos, o qual foi analisado e aceito. A equipe do CPqD então elaborou um documento compreendendo informações de nível mais alto sobre o projeto, tais como o detalhamento dos requisitos funcionais e identificação de riscos. Em seguida foi elaborado um outro documento que contém informações detalhadas do projeto e de tudo o que foi implementado no FPGA (instruções de configuração, mapa de registradores, arquiteturas, nomes e descrições dos principais blocos, sub-blocos e sinais, domínios de relógio, entre outras infomações). O aceite final se deu pelas demonstrações em laboratório feitas ao

cliente e pela aprovação de um terceiro documento elaborado pela equipe do CPqD, o qual contém informações sobre os testes e seus resultados e também sobre como cada requisito foi atendido.

1.3 Contribuição

Esta dissertação discorre sobre dois códigos EFEC. O primeiro deles, doravante referido como **EFEC I.9 original**, é um dos códigos propostos na recomendação G.975.1 do ITU-T [3], mais especificamente aquele localizado no anexo I.9. Trata-se de um código EFEC composto por dois códigos BCH(1020,988) (Bose-Chaudhuri-Hocquenghen) entrelaçados e com decodificação iterativa. Segundo a recomendação G.975.1, tal código concede NCG superior a 8,5 dB à BER_{out} de 10^{-13} para uma decodificação com 10 iterações e um dos motivos que levaram à escolha do mesmo para o projeto foi justamente o seu alto desempenho. O outro motivo descende do fato de suas características comportarem modificações para aumentar a quantidade de redundância, no intuito de obter um segundo código, doravante denominado **EFEC I.9 modificado**, capaz de prover NCG mais próximo dos 10 dB requeridos pela Padtec. Assim como o EFEC I.9 original, o EFEC I.9 modificado também é composto por dois BCH(1020,988) entrelaçados e dispõe de decodificação iterativa. Ambos são abordados e comparados neste documento.

A combinação entre a implementação dos códigos supracitados em FPGA e os trabalhos de pesquisa e escrita desta dissertação resulta em várias contribuições, dentre as quais podem ser citadas como mais relevantes:

- As arquiteturas de alto nível.

  Para facilitar o entendimento e a implementação em FPGA dos códigos EFEC tema desta dissertação, foi necessário desenvolver arquiteturas de alto nível para os codificadores e decodificadores, as quais são apresentadas sob a forma de diagrama de bloco e também detalhadamente explicadas no Capítulo 5.

- Dados de desempenho.

  O código EFEC I.9 modificado foi avaliado em laboratório por meio da medição das taxas de erro de saída BER_{out} que o mesmo propiciava para determinados valores de taxa de erro de entrada BER_{in}. Os valores medidos foram usados para cálculo de dados de desempenho como ganho de codificação (CG), ganho de codificação líquido

(NCG) e fator Q e são apresentados sob a forma de tabela no Capítulo 6. O mesmo capítulo apresenta ainda as curvas de BER_{in} *versus* BER_{out} e BER *versus* fator Q para o EFEC I.9 modificado, acompanhadas das respectivas curvas de desempenho para o EFEC I.9 original e para o GFEC que constam na recomendação G.975.1.

- Dados resultantes das implementações em FPGA dos EFECs.
  As frequências máximas de operação dos codificadores e decodificadores dos EFECs I.9 original e modificado são apresentadas no Capítulo 6 juntamente com dados relacionados ao consumo de recursos lógicos do FPGA demandados por cada um, tais como número de registradores e tabelas de busca adaptativas (ALUT, *adaptative look-up table*).

- Um artigo.
  O artigo intitulado "Ampliação de Ganho de Codificação em um Código EFEC BCH para Comunicações Ópticas" foi apresentado no Simpósio Brasileiro de Telecomunicações (SBrT), um dos eventos mais importantes do país na área de telecomunicações.

- Seis pedidos de patente depositados no Instituto Nacional de Propriedade Industrial (INPI).

  BR 10 2014 003615-6 [4]: Arquitetura e Método Para Cálculo dos Vetores q em Códigos EFEC. A invenção tem como principal objetivo prover métodos e arquiteturas de circuitos para a implementação do cálculo da paridade extra ou vetores q dos códigos EFEC I.9 (explicado na Seção 5.2.4), de modo que seja possível atender a diferentes requisitos de latência e estabelecer uma relação de compromisso entre a área ocupada (recursos lógicos) e latência total. A invenção deve permitir que se tenha desde um alto aproveitamento de área em detrimento da latência até alta capacidade de processamento (com baixa latência) mediante um acréscimo de área que seja pouco relevante perante o tamanho total do circuito; ou ainda combinações intermediárias onde o requisito de latência é suficientemente atendido e a área é otimizada.

  BR 10 2014 022170-0 [5]: Método de Aceleração para Decodificação Iterativa de Códigos Corretores de Erro e Arquitetura de Decodificador Iterativo. Este método se aplica no estágio de decodificação iterativa do código EFEC I.9, no qual dois tipos de decodificadores operam de forma alternada sobre um mesmo bloco de dados a ser corrigido. Quando a correção de erros é executada por um circuito convencional,

existe um limite natural no número de iterações. Esta patente propõe um método para ultrapassar tal limite por meio de um pré-processamento dos dados a serem decodificados aumentando, assim, a eficiência na correção de erros.

BR 10 2013 026036-3 [6]: Arquitetura de Circuito e Método para Entrelaçamento de *Bits* em Fluxo de Dados Paralelo. A invenção em questão tem como principal objetivo prover um método de alta eficiência, ou seja, de baixa latência e com um circuito reduzido, para fazer o entrelaçamento de *bits* em um fluxo de dados paralelo de entrada e saída, mantendo a taxa de dados de saída igual à de entrada. Outro objetivo é prover um método de entrelaçamento de *bits* que atenda à regra definida na Recomendação G.975.1, Anexo I.9.

BR 10 2014 027625-4 [7]: Arquitetura de Circuito e Método para Processamento Recursivo de Dados Acessados de Memória sem Penalidade de Latência. A ideia em questão resolve um problema crítico de tempo de acesso aos dados armazenados em memória, nos casos em que estes dados são acessados sequencialmente e particularmente quando uma mesma posição de memória é acessada mais de uma vez em um intervalo de tempo muito curto. A patente propõe um circuito adicional para acesso à memória que supre imediatamente o processador quando o mesmo requisitar a leitura de um dado recém modificado. Dessa forma, a latência existente em um circuito convencional é eliminada e a capacidade de processamento é aumentada.

BR 10 2013 019299-6 [8]: Circuito e Método Gerador de Pulso Síncrono Disparado por Comando Proveniente de Outro Domínio de Relógio. O principal objetivo dessa invenção é prover um método capaz de gerar um pulso de um único período e síncrono com o relógio de destino, por intermédio de um comando gerado em domínio de relógio independente (diferente), sem gerar falhas ou instabilidade.

BR 10 2013 031832-9 [9]: Circuito e Método para Armazenamento Síncrono de uma Palavra Digital com Transferência Automática para Outro Domínio de Relógio. Trata-se de um circuito que pode ser usado em sistemas digitais para garantir que uma informação formada por um ou mais dígitos binários (compondo uma única informação) possa ser transferida com segurança (sem alterar a informação) entre domínios de relógio que diferem em fase, frequência ou ambos. O circuito proposto se baseia no depósito da informação na entrada do circuito, controlado por meio de um sinal indicador de escrita.

- Vinte e um pedidos de registro de programa de computador depositados no Instituto Nacional de Propriedade Industrial (INPI).

  Dos vinte e um pedidos em questão, oito são relacionados à implementação em lógica programável dos codificadores EFEC I.9, onze são relacionados à implementação em lógica programável dos decodificadores EFEC I.9, um está ligado à implementação das interfaces de comando de linha (CLI, *command line interface*) e de programação de aplicação (API, *application programm interface*) para trocar informações com o processador externo e o último registro corresponde à implementação de rotinas de testes.

1.4 Organização da Dissertação

O Capítulo 2 mostra um sistema de comunicação e os principais blocos funcionais que o integram. Motivos que levam à degradação dos dados em tais sistemas também são descritos nesse capítulo.

O Capítulo 3 versa sobre vários aspectos abordados em teoria de informação e codificação, desde princípios básicos de álgebra abstrata e campos de Galois até a classificação dos códigos corretores de erro.

O Capítulo 4 introduz informações do protocolo OTN necessárias para o entendimento deste trabalho.

O Capítulo 5 apresenta, baseado na recomendação G.975.1 do ITU-T [3], as principais características do código EFEC I.9, tema deste trabalho. Além disso, são mostrados dados das arquiteturas criadas para a implementação dos EFECs I.9 original e modificado.

No Capítulo 6 é mostrado o ambiente montado em laboratório para testar as implementações citadas no Capítulo 5. Os principais resultados também são apresentados e discutidos nesse capítulo.

Finalmente, no Capítulo 7 são apresentados conclusões e projetos futuros.

Capítulo 2

Sistemas de Comunicação

2.1 Modelo Básico de um Sistema de Comunicação

Um sistema de comunicação envolve a transmissão de informação a partir de um ponto de origem até outro onde ela será consumida. Independentemente do tipo de sistema de comunicação considerado, sempre existem três elementos básicos: o **transmissor** (origem), o **receptor** (destino) e o **canal**, que conecta os dois primeiros elementos. Durante a transmissão e a recepção, a informação passa por uma sucessão de blocos com funções específicas, ilustrados na Figura 2.1 e comentados em seguida.

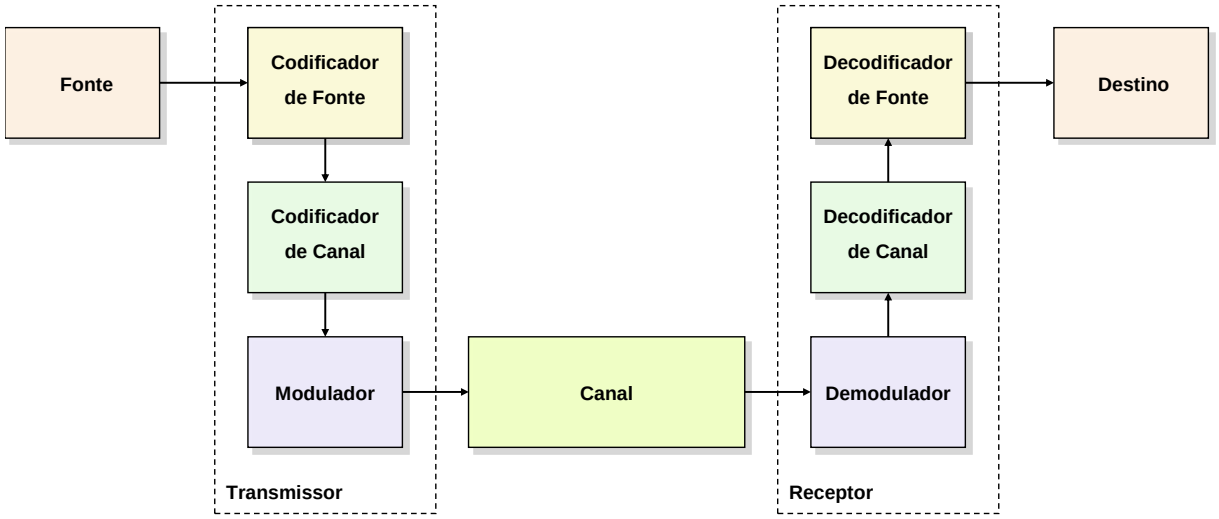

Figura 2.1: Diagrama de blocos funcionais de um sistema de comunicação.

No primeiro dos blocos funcionais localizados do lado do **transmissor**, a **fonte**, a informação é produzida. No **codificador de fonte**, símbolos binários são atribuídos à

informação produzida na fonte, de modo que as redundâncias desnecessárias sejam reduzidas e a eficiência aumente. O **codificador de canal**, por sua vez, insere informação redundante e controlada aos símbolos gerados pelo codificador de fonte, possibilitando ao receptor usar tais informações para estimar um sinal recebido mais próximo daquele que foi enviado, mesmo que este tenha sofrido efeitos degradantes causados pelo canal. Antes de ser levado ao canal, o sinal passa pelo último bloco funcional do transmissor: o **modulador**; cuja função é adequar o sinal de informação para ser transmitido, segundo as características do canal. Como existem vários tipos de modulação, deve ser escolhida aquela que melhor atenda as características do projeto, como por exemplo: banda disponível, potência de transmissão, tipos de ruído e interferência tipicamente encontrados no canal, etc.

O **receptor** tem por função recuperar a informação enviada a partir do sinal recebido. Para tal, os processamentos que foram realizados no transmissor são revertidos, de tal sorte que a estimativa de informação entregue ao **destino** seja tão próxima quanto possível do que foi gerado pela fonte.

Conforme já foi comentado anteriormente, o **canal** é o meio físico através do qual a informação é transmitida da fonte ao destino. Nas transmissões com meio guiado (com fio), os sinais elétricos trafegam pelo material condutor (tipicamente cobre) ou são convertidos em sinais luminosos para trafegar por uma fibra óptica. Já nas transmissões não guiadas (sem fio), a transmissão é feita por irradiação eletromagnética, irradiando-se a informação pelo ar com o uso de antenas.

2.2 Degradação dos Dados em Sistemas de Comunicação

Nem sempre o sinal recebido é igual àquele que foi transmitido. Todos os sistemas estão sujeitos a fenômenos que poderão degradar a informação original, limitando seu desempenho. Por exemplo, um canal poderá modificar a frequência, a fase e a amplitude do sinal transmitido de maneira diferente de outro, pois ambos possuem características e susceptibilidade peculiares a esses fenômenos. Dá-se o nome de distorção às alterações na forma de onda de um sinal obtido na saída de um sistema em relação àquele aplicado na entrada.

Para que um canal não cause distorções em um sinal $x(t)$ aplicado à sua entrada, o sinal de saída $y(t)$ deve ser uma réplica de $x(t)$, multiplicado por uma constante de ganho K e deslocado no tempo de t_d, porém mantendo sua forma inalterada, ou seja,

$$y(t) = Kx(t - t_d), \tag{2.1}$$

onde K é uma constante maior que zero e t_d é o atraso. Aplicando-se a transformada de Fourier em ambos os lados da Equação (2.1), vem

$$Y(\omega) = Ke^{-j\omega t_d}X(\omega). \tag{2.2}$$

Se o canal em questão for considerado um sistema linear e invariante no tempo (LIT), é possível demonstrar que a saída $y(t)$ é igual à convolução da entrada $x(t)$ com a resposta ao impulso do sistema $h(t)$. Isto é,

$$y(t) = x(t) * h(t). \tag{2.3}$$

Aplicando-se a transformada de Fourier em ambos os lados da Equação (2.3), tem-se

$$Y(\omega) = X(\omega)H(\omega). \tag{2.4}$$

Da Equação (2.4) obtém-se que a resposta em frequência $H(\omega)$ do sistema é igual a

$$H(\omega) = \frac{Y(\omega)}{X(\omega)}. \tag{2.5}$$

Seja

$$H(\omega) = |H(\omega)|e^{j\theta_H(\omega)}. \tag{2.6}$$

Então $|H(\omega)|$ é a resposta de amplitude ou resposta de magnitude do sistema, enquanto $\theta_H(\omega)$ é a resposta de fase do sistema. Portanto, da Equação (2.4) conclui-se que para uma transmissão sem distorção, o sistema deve ter

$$H(\omega) = |H(\omega)|e^{j\theta_H(\omega)} = Ke^{-j\omega t_d}. \tag{2.7}$$

Sendo assim, $|H(\omega)| = K$ e $\theta_H(\omega) = -j\omega t_d$, ou seja, a amplitude de $H(\omega)$ deve ser

constante em todo o espectro de frequências de interesse e a fase de $H(\omega)$ deve ser linear com a frequência.

Segundo [10], um sinal de dados pode sofrer distorções de dois tipos ao passar por um canal: lineares e não lineares.

2.2.1 Distorções Lineares

As distorções lineares ocorrem quando um canal (ou um sistema) provoca modificações na amplitude e ou na fase de um sinal que trafega por ele, de modo que não haja adição de novas componentes espectrais ao sinal. Este tipo de distorção é previsível, uma vez que se conhece o comportamento do canal. Em vista disso, ela pode ser descrita através de uma função de transferência e seus efeitos podem ser reduzidos a níveis toleráveis ou praticamente eliminados por meio do uso de sistemas lineares, chamados de redes de equalização ou simplesmente de equalizadores, que são colocados em cascata com o canal. Tais redes devem ser projetadas de tal sorte que a cascata das funções de transferência do canal $H_C(\omega)$ e da rede de equalização $H_{EQ}(\omega)$ torne verdadeira a equação

$$H_C(\omega).H_{EQ}(\omega) = Ke^{-j\omega t_d}. \tag{2.8}$$

2.2.1.1 Distorção de Amplitude

Independentemente da frequência, a resposta de magnitude de um canal deve ser constante (ao menos na faixa de frequência onde está o espectro do sinal a ser transmitido) para que as componentes de frequência desse sinal sejam multiplicadas pela mesma constante K. Todavia, se $|H(\omega)|$ não for constante em todo o espectro de frequências, ocorrerá distorção de amplitude, fazendo com que a forma do sinal de saída seja igual à do de entrada, porém com ganhos e atenuações em algumas frequências. Este tipo de distorção pode ser causado, por exemplo, por filtros, cabos, transformadores ou capacitores presentes na linha de transmissão.

2.2.1.2 Distorção de Fase

Se o desvio de fase introduzido pelo canal variar linearmente com a frequência, o sinal não será distorcido e sim simplesmente atrasado. Entretanto, se a resposta de fase $\theta_H(\omega)$ não for linear com a frequência ao menos dentro da faixa ocupada pelo sinal, haverá dis-

torção de fase. Dito fenômeno acarretará mudanças na forma de onda do sinal de saída em relação à do sinal de entrada, pois atrasos diferentes afetarão as componentes espectrais de frequência do sinal.

2.2.2 Distorções Não Lineares

As distorções não lineares são fenômenos de natureza aleatória que alteram características do sinal original, dificultando a correta decodificação da informação pelo receptor. Esse tipo de distorção pode tanto dar origem a harmônicos ou outras componentes de frequência que não estavam presentes na informação original quanto suprimir aquelas que estavam. Podem ser introduzidas, por exemplo, por elementos não lineares presentes em um sistema, tais como bobinas. A existência de não linearidades implica a não existência de uma função de transferência.

2.2.3 Ruídos

Entende-se por ruído um sinal indesejado, completamente aleatório tanto em amplitude quanto em fase, que altera a informação original e dificulta sua detecção pelo receptor. De acordo com [10], o maior causador de erros em uma transmissão de dados é o ruído branco ou ruído térmico presente no canal. Sua origem se deve à agitação natural aleatória dos elétrons presentes em seus elementos condutores e ele é diretamente proporcional à temperatura, pois a partir do zero absoluto (0 kelvin) quanto maior for a temperatura, mais intenso será o movimento aleatório dos elétrons livres, gerando pequenas correntes elétricas no condutor.

Quando o ruído térmico é a principal causa de alterações no sinal original, é comum a utilização de um modelo simplificado para caracterizar estatisticamente um canal. Neste caso, apoiado no teorema do limite central, o ruído é modelado como um processo gaussiano, que se soma ao sinal original, levando-o a ser chamado de ruído aditivo gaussiano branco (AWGN, *additive white gaussian noise*).

Outro tipo de ruído importante é o chamado ruído balístico (*shot noise*). Também conhecido como ruído de Poisson, ele aparece em componentes como diodos e transistores e resulta do fato de a corrente nesses componentes não ser um fluxo contínuo, mas sim a soma de pulsos discretos no tempo, cada um correspondendo a uma transferência de elétrons através do condutor. Por exemplo, em um circuito fotodetector um pulso de

corrente é gerado toda vez que um elétron é emitido pelo catodo devido à incidência de luz a partir de uma fonte de intensidade constante. A corrente total que flui pelo fotodetector pode ser modelada por um processo estacionário $X(t)$ denominado ruído balístico, que consiste na soma infinita de pulsos de corrente mostrado na expressão

$$X(t) = \sum_{k=-\infty}^{\infty} h(t - \tau_k), \tag{2.9}$$

onde $h(t - \tau_k)$ é o pulso de corrente gerado no instante τ_k.

Há ainda outros tipos de ruído que podem afetar um sistema de comunicação, dos quais podem ser citados como exemplos o ruído atmosférico e os ruídos causados pelo homem (*man-made noises*).

2.2.4 Degradação de Sinal em Sistemas de Comunicação Óptica

O trabalho que deu origem a essa dissertação tem como foco a correção de erros em sinais que se propagam por fibras ópticas. Assim sendo, é conveniente abordar as principais causas de degradação de sinal que afetam esse tipo de canal.

2.2.4.1 Atenuação

Ao se propagar por um meio diferente do vácuo, um sinal invariavelmente sofrerá atenuação e a fibra óptica não é exceção a essa regra. Embora as fibras de sílica possibilitem transmitir informação sob a forma de luz com baixa perda (da ordem de 0,2 dB/km), após 100 quilômetros a potência óptica é reduzida a apenas 1% da original [11]. Portanto, a atenuação é um fator determinante na distância máxima entre o transmissor e o receptor ou entre o transmissor e o amplificador e deve ser levada em conta no projeto de enlaces ópticos [12].

O coeficiente de atenuação da fibra óptica é representado pela letra grega α e normalmente é expressado em decibéis por quilômetro, ou dB/km, assim

$$\alpha[dB/km] = \frac{10}{z} \log \frac{P(0)}{P(z)}, \tag{2.10}$$

onde $P(0)$ é a potência óptica na origem e $P(z)$ é a potência óptica na distância z em

relação à origem. Também chamado de perda da fibra, o coeficiente de atenuação da fibra depende de alguns fatores que serão expostos a seguir [13].

- Absorção Material

  A absorção material subdivide-se em dois tipos: a intrínseca, que é relacionada à ressonância entre o material base da fibra (por exemplo a sílica SiO_2) e a onda eletromagnética que transporta o sinal. Já a absorção material extrínseca é devida à presença de outros materiais na fibra, tais como OH^- (hidroxila), ferro, cobre, cromo e vanádio. Atualmente existem modernas técnicas de produção que reduzem os níveis de impureza da fibra, permitindo a fabricação das chamadas fibras de baixa perda.

- Espalhamento Rayleigh

  O espalhamento Rayleigh resulta do processo produtivo da fibra e é causado por microvariações na densidade do material da mesma. Quando a luz se propaga por uma região onde há variações no índice de refração em distâncias pequenas, quando comparadas ao comprimento de onda do sinal, há um espalhamento da luz chamado de espalhamento Rayleigh. Esse fenômeno é o mesmo que espalha a luz do sol na atmosfera e faz com que o céu tenha a cor azul durante o dia.

Outros fatores também devem ser levados em consideração no projeto de um sistema de comunicações ópticas, pois causam atenuação de potência óptica. São eles: perdas em emendas e conectores, perda por inserção em componentes como filtros, acopladores, circuladores, etc.

2.2.4.2 Dispersão

De acordo com [14], dá-se o nome de dispersão aos efeitos que derivam do fato de que as componentes de um sinal trafegam em diferentes velocidades na fibra e, por isso, chegam em diferentes momentos ao receptor. Esse fenômeno é cumulativo (quanto mais longo o enlace, mais distorção) e provoca o alargamento do pulso de informação em comparação com aquele que foi transmitido originalmente, causando interferência intersimbólica (ISI, *intersymbol inteference*). A ISI faz com que um símbolo transmitido interfira no sinal do símbolo subsequente como se fosse um ruído, comprometendo a capacidade do receptor de distinguir corretamente a informação e diminuindo o alcance do enlace [15].

Os principais tipos de dispersão em comunicações ópticas são:

- Dispersão Intermodal

  Também conhecida como dispersão multimodal, ocorre nas fibras ópticas multimodo e resulta do fato de cada um dos modos possuir um valor de velocidade de grupo para uma dada frequência.

- Dispersão Intramodal

  A dispersão cromática, como também é chadada a dispersão intramodal na literatura, ocorre nas fibras ópticas monomodo em decorrência do fato de que as componentes espectrais trafegam com velocidades de grupo ligeiramente diferentes, conforme o índice de refração. Essa relação é dada por

$$v_p(\omega) = \frac{c}{n + n(\omega)}, \tag{2.11}$$

  onde $v_p(\omega)$ é a velocidade de fase de uma componente espectral e n é o índice de refração do material da fibra e ω é a frequência angular em radianos por segundo $[rad/s]$.

Existem algumas formas para combater os efeitos da dispersão em comunicações ópticas, dentre as quais estão: o emprego de fibras modificadas chamadas de fibras compensadoras de dispersão (DCF, *dispersion compensating fiber*) e também o emprego de redes de Bragg (FBG, *fiber Bragg grating*) que, segundo [16], são segmentos de redes de difração cujos índices de refração do núcleo são modificados para que se tornem um pouco maiores que o original e que atuam como filtros, fazendo com que alguns comprimentos de onda do espectro da luz sejam refletidos ao incidir nessas estruturas, enquanto outros sejam transmitidos sem perdas. Outra maneira de combater os efeitos da dispersão é aumentando a robustez do sistema mediante o emprego de códigos corretores de erro.

Capítulo 3

Fundamentos Matemáticos de Códigos Corretores de Erro

O estudo de códigos corretores de erro envolve conhecimentos de várias áreas, mas a álgebra abstrata e a teoria de Galois constituem áreas da matemática de fundamental importância no estudo desse tipo de código. Neste capítulo serão apresentados conceitos básicos de álgebra abstrata necessários ao entendimento da teoria de Galois. Serão apresentados também alguns exemplos de códigos, incluindo o BCH.

3.1 Álgebra Abstrata

3.1.1 Conjuntos

Denomina-se **conjunto** uma coleção de objetos quaisquer, sendo que tais objetos são chamados elementos [17]. Os conjuntos $C_0 = \{0, 2\}$ e $C_1 = \{banana, uva, laranja\}$ são exemplos de conjuntos finitos, enquanto o conjunto dos números inteiros $\mathbb{Z}$ é um exemplo de conjunto infinito.

3.1.2 Grupos

Seja G um conjunto de elementos e $*$ uma operação binária que relaciona dois elementos a e b desse conjunto, de tal sorte que um terceiro elemento c, também pertencente a G, é atribuído como resultado da operação $a * b$. O conjunto G é classificado como **grupo** $(G, *)$ se as seguintes condições forem satisfeitas [18]:

1. A operação binária $*$ é associativa, ou seja
   $a * (b * c) = (a * b) * c$.

2. O conjunto G contém um elemento identidade e, tal que
   $e * a = a * e = a$.

3. Para cada elemento de G existe um elemento a' também pertencente a G, chamado de elemento inverso de a, de modo que
   $a * a' = a' * a = e$.

Se além das condições acima o grupo também satisfizer a condição $a*b = b*a, \forall a, b \in G$, ele é classificado como comutativo ou abeliano.

3.1.3 Anéis

Segundo [17], um conjunto A não vazio acompanhado de duas operações binárias $+$ e $\cdot$, chamadas respectivamente de adição e multiplicação, é classificado como **anel** $(A, +, \cdot)$ se:

1. A adição e a multiplicação são associativas $\forall a, b, c \in A$. Ou seja
   $a + (b + c) = (a + b) + c$
   $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.

2. A adição e a multiplicação são comutativas $\forall a, b \in A$.
   $a + b = b + a$
   $a \cdot b = b \cdot a$

3. Existe um elemento neutro para a adição e outro para a multiplicação, de modo que
   $\exists\ e \in A \mid a + e = a,\ \forall a \in A$.
   $\exists\ f \in A, f \neq e, \mid a \cdot f = a,\ \forall a \in A$.

4. Em relação à adição, existe um elemento inverso a' para todo elemento de A.
   $\exists\ a' \in A \mid a + a' = e,\ \forall a \in A$

5. A multiplicação é distributiva em relação à adição, tal que
   $a \cdot (b + c) = a \cdot b + a \cdot c,\ \forall a, b, c \in A$.

3.1.4 Campos

Seja F um conjunto não vazio tal que as operações binárias adição e multiplicação, juntamente com suas respectivas inversas (subtração e divisão), quando são realizadas entre seus elementos, resultam em outro elemento também pertencente a F. De acordo com [18], esse conjunto será classificado como um **campo** se:

1. F for um grupo abeliano com relação à adição.

   O elemento identidade para a adição é o "0", pois $0 + a = a + 0 = a$.

2. F for um grupo abeliano com relação à multiplicação.

   O elemento identidade para a multiplicação é o "1", pois $1 \cdot a = a \cdot 1 = a$.

3. A multiplicação for distributiva em relação à adição.

   $a \cdot (b + c) = a \cdot b + a \cdot c$

3.1.4.1 Campos de Galois

Os campos que possuem um número finito de elementos são chamados de campos finitos ou **campos de Galois** [18], em homenagem a seu descobridor Évariste Galois (1811-1832). São representados pela sigla $GF(q)$, onde q é uma potência de um número primo que reflete o número de elementos do campo e determina sua ordem.

Para todo número primo p é possível construir um campo $GF(p)$ cujos elementos são $\{0, 1, 2, ..., p - 1\}$ e cujas operações são a soma e o produto módulo p. Considerando-se o caso em que p vale 2, tem-se o conjunto $\{0, 1\}$ que, junto com as operações soma e multiplicação modulo 2 mostradas na Tabela 3.1, forma um campo com dois elementos chamado **campo de Galois binário** ou $GF(2)$ [19] que, apesar de ser o mais simples dos campos de Galois, tem um papel muito importante em teoria de códigos, dada a natureza binária das comunicações digitais.

Tabela 3.1: Adição e multiplicação módulo 2 em $GF(2)$.

$\oplus$	0	1
0	0	1
1	1	0

$\odot$	0	1
0	0	0
1	0	1

A aritmética módulo n pode ser realizada de maneira simples, bastando para isso calcular normalmente a adição ou a multiplicação e, em seguida, dividir o resultado por n

para levá-lo novamente a um valor que pertence ao campo. Por exemplo: supondo-se um conjunto $\{0, 1, 2\}$, a operação $2 \oplus 1$ fica

$$2 \oplus 1 = 3$$
$$3 \bmod 3 = 0.$$

Já a operação $2 \odot 2$ fica

$$2 \odot 2 = 4$$
$$4 \bmod 3 = 1.$$

Assim, as operações de adição e multiplicação com os elementos desse conjunto ficam como mostrado na Tabela 3.2.

Tabela 3.2: Adição e multiplicação módulo 3 para um alfabeto de 3 símbolos.

$\oplus$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\odot$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Os elementos de um campo também são comumente representados sob a forma polinomial. Um polinômio sobre um corpo $GF(q)$ é uma expressão matemática da forma

$$f(x) = f_{n-1}x^{n-1} + f_{n-2}x^{n-2} + \ldots + f_1x^1 + f_0, \tag{3.1}$$

onde os coeficientes f_0, f_1, ..., f_{n-1}, assumirão os valores 0 ou 1 para o caso do campo binário. Um número binário de n *bits* pode ser representado por um polinômio de grau $n-1$. Portanto, o número binário 100010101 pode ser representado pelo polinômio

$$f(x) = x^8 + x^4 + x^2 + 1. \tag{3.2}$$

As operações de adição e multiplicação podem ser realizadas na forma polinomial usando-se a aritmética de módulo 2 e também as propriedades comutativa, distributiva e associativa. A divisão entre dois polinômios também vale e é uma importante ferramenta. Tal operação tem a forma

$$f(x) = q(x)g(x) + r(x), \tag{3.3}$$

onde $g(x)$ é o polinômio divisor, $q(x)$ é o polinômio quociente e $r(x)$ é o resto da operação. Exemplo: Ao se dividir o polinômio x^3+x+1 por $x+1$, obtém-se $q(x) = x^2+x$ e $r(x) = 1$.

Como citado anteriormente, o número de elementos de um campo de Galois é igual a q. Ocorre que, para que se forme o campo $GF(2^m)$, q deve ser uma potência p^m onde p é um número primo e m é um número inteiro positivo. No caso do campo de Galois binário p é igual a 2 e m vale 1, mas é possível estender o conceito escolhendo-se outros valores para p e m apropriadamente. Campos $GF(2^m)$ são extensões do campo binário que possuem 2^m elementos de m bits e cujos elementos podem ser gerados a partir de um polinômio $p(x)$, de grau m, irredutível (não fatorável) e primitivo, ou seja, o menor inteiro positivo n para o qual $p(x)$ divide $x^n + 1$ é $n = 2^m - 1$. Por exemplo: o polinômio $p(x) = x^4 + x + 1$ divide $x^{15} + 1$, mas não divide outro $x^n + 1$ para $1 \leq n < 15$. Por essa razão, $x^4 + x + 1$ é um polinômio primitivo. Existem tabelas que listam polinômios desse tipo [20], pois sua obtenção não é simples. A Tabela 3.3 lista os principais polinômios primitivos de grau m utilizados para geração de extensões de campos de Galois.

Tabela 3.3: Principais polinômios primitivos de grau m.

m	**Polinômio Primitivo**	m	**Polinômio Primitivo**
3	$x^3 + x + 1$	14	$x^{14} + x^{10} + x^6 + x + 1$
4	$x^4 + x + 1$	15	$x^{15} + x + 1$
5	$x^5 + x^2 + 1$	16	$x^{16} + x^{12} + x^3 + x + 1$
6	$x^6 + x + 1$	17	$x^{17} + x^3 + 1$
7	$x^7 + x^3 + 1$	18	$x^{18} + x^7 + 1$
8	$x^8 + x^4 + x^3 + x^2 + 1$	19	$x^{19} + x^5 + x^2 + x + 1$
9	$x^9 + x^4 + 1$	20	$x^{20} + x^3 + 1$
10	$x^{10} + x^3 + 1$	21	$x^{21} + x^2 + 1$
11	$x^{11}x^2 + 1$	22	$x^{22} + x + 1$
12	$x^{12} + x^6 + x^4 + x + 1$	23	$x^{23} + x^5 + 1$
13	$x^{15} + x^4 + x^3 + x + 1$	24	$x^{24} + x^7 + x^2 + x + 1$

Considerando-se os elementos 0 e 1 do campo de Galois binário, a operação multiplicação $\cdot$ e supondo que α seja um terceiro elemento não nulo, vem [19]:

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 1 = 1$$

$$0 \cdot \alpha = \alpha \cdot 0 = 0$$

$$1 \cdot \alpha = \alpha \cdot 1 = \alpha$$

$$\alpha^2 = \alpha \cdot \alpha$$

$$\alpha^3 = \alpha \cdot \alpha \cdot \alpha$$

$$\alpha^4 = \alpha \cdot \alpha \cdot \alpha \cdot \alpha$$

$$\vdots$$

$$\alpha^j = \alpha \cdot \alpha \cdot \cdots \cdot \alpha \ (j \text{ vezes}).$$

Portanto

$$0 \cdot \alpha^j = \alpha^j \cdot 0 = 0$$

$$1 \cdot \alpha^j = \alpha^j \cdot 1 = \alpha^j$$

$$\alpha^i \cdot \alpha^j = \alpha^j \cdot \alpha^i = \alpha^{i+j}.$$

Pelas considerações acima, define-se o seguinte conjunto F sobre o qual a operação $\cdot$ é definida

$$F = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \ldots, \alpha^j, \ldots\}.$$

Considerando agora que $p(x)$ é um polinômio primitivo e que α é uma raiz desse polinômio, ou seja $p(\alpha) = 0$, então $p(x)$ divide $x^{2^m-1} + 1$. Portanto

$$x^{2^m-1} + 1 = q(x) \cdot p(x). \tag{3.4}$$

Substituindo-se x por α na Equação (3.4), vem

$$\alpha^{2^m-1} + 1 = q(\alpha) \cdot p(\alpha) \rightarrow \alpha^{2^m-1} + 1 = q(\alpha) \cdot 0 \rightarrow \alpha^{2^m-1} + 1 = 0.$$

Então

$$\alpha^{2^m-1} + 1 = \alpha^0 \tag{3.5}$$

A Equação (3.5) mostra que existe um elemento $\alpha^{2^m-1} \neq 0$ a partir do qual os elementos do conjunto F se repetem. Assim, conclui-se que F será finito e conterá os elementos

$$F = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \ldots, \alpha^{2^m-2}\}. \tag{3.6}$$

Como exemplo, suponha que se queira criar uma extensão do campo de Galois binário $GF(2)$ com m igual a 4, ou seja, $GF(2^4)$. Suponha também que o polinômio primitivo escolhido seja $1 + x + x^4$. Se α é uma raiz desse polinômio, então $p(\alpha) = 0$ e

$$1 + \alpha + \alpha^4 = 0 \rightarrow \alpha^4 = 1 + \alpha.$$

Por meio de substituições, é possível encontrar os demais elementos do campo. Por exemplo,

$$\alpha^5 = \alpha \cdot \alpha^4 = \alpha \cdot (\alpha + 1) = \alpha + \alpha^2$$
$$\alpha^6 = \alpha \cdot \alpha^5 = \alpha \cdot (\alpha + \alpha^2) = \alpha^2 + \alpha^3$$
$$\vdots$$
$$\alpha^{2^m-2} = \alpha^{14} = 1 + \alpha^3.$$

A Tabela 3.4 mostra todos os 16 elementos desse campo, que podem ser representados sob as seguintes formas: potências de α, polinomial ou binária.

Tabela 3.4: Representações dos elementos em $GF(2^4)$ gerados por $p(x) = 1 + x + x^4$.

Potências de α	Polinomial	Binária α^0,α^1,α^2,α^3
0	0	0000
α^0	1	1000
α^1	α	0100
α^2	α^2	0010
α^3	α^3	0001
α^4	$1 + \alpha$	1100
α^5	$\alpha + \alpha^2$	0110
α^6	$\alpha^2 + \alpha^3$	0011
α^7	$1 + \alpha + \alpha^3$	1101
α^8	$1 + \alpha^2$	1010
α^9	$\alpha + \alpha^3$	0101
α^{10}	$1 + \alpha + \alpha^2$	1110
α^{11}	$\alpha + \alpha^2 + \alpha^3$	0111
α^{12}	$1 + \alpha + \alpha^2 + \alpha^3$	1111
α^{13}	$1 + \alpha^2 + \alpha^3$	1011
α^{14}	$1 + \alpha^3$	1001

3.1.5 Espaços Vetoriais Sobre o Campo Binário

Uma sequência de símbolos binários $v = (v_1, v_2, ..., v_n)$ pode ser definida como uma ênupla sobre $GF(2)$. Existem, portanto, 2^n ênuplas diferentes, onde o valor de v_i pode ser

os valores binários 0 ou 1. Um **espaço vetorial** V_n sobre o campo $GF(2)$ é o conjunto de todas as ênuplas binárias, agora denominadas vetores, para as quais se aplicam as operações binárias adição e multiplicação, cujos símbolos são respectivamente $\oplus$ e $\odot$, mostradas na Tabela 3.1. Por exemplo, para os vetores $v = (v_1, v_2, ..., v_n)$ e $u = (u_1, u_2, ..., u_n)$, a operação $v \oplus u$ resulta

$$v \oplus u = (v_1 \oplus u_1, v_2 \oplus u_2, ..., v_n \oplus u_n). \tag{3.7}$$

Já operação binária $v \odot u$ resulta

$$v \odot u = (v_1 \odot u_1, v_2 \odot u_2, ..., v_n \odot u_n). \tag{3.8}$$

O espaço vetorial V_4 é mostrado na Tabela 3.5. A adição de qualquer vetor de V_4 resulta em outro vetor também pertencente a V_4.

Tabela 3.5: Espaço vetorial V_4 sobre $GF(2)$.

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

Para cada vetor de V_4 existem apenas duas multiplicações por escalares. Por exemplo, para o vetor 1010, as multiplicações por escalares são

$$0 \odot 1010 = 0000 \tag{3.9}$$

$$1 \odot 1010 = 1010. \tag{3.10}$$

3.1.5.1 Subespaço Vetorial

Dentro de um espaço vetorial V é possível encontrar um conjunto de vetores S, não vazio, que obedece todas as condições para ser um espaço vetorial. O conjunto S será um **subespaço vetorial** de V se a soma de qualquer par de vetores pertencente a S resultar em outro vetor também pertencente a S e se qualquer elemento do campo sobre o qual se apoia o espaço vetorial V (o que no caso de $GF(2)$ significa os elementos binários 0 e 1) multiplicado por qualquer vetor de S resultar em outro vetor também pertencente a S [18].

Um exemplo de subespaço vetorial de V_4 é

$$S = \{0000, 1001, 0100, 1101\}. \tag{3.11}$$

3.2 Códigos Corretores de Erro

Conforme já foi mencionado anteriormente, os códigos corretores de erro adicionam redundância controlada à informação que será transmitida, tornando possível ao receptor detectar e corrigir eventuais erros que podem ter sido inseridos na informação original durante a transmissão. Duas importantes classes de códigos são comentadas a seguir.

3.2.1 Códigos de Bloco

De acordo com [21], códigos em que o processo de codificação se dá sobre blocos de *bits* com comprimento fixo igual a k recebem o nome de **códigos de bloco**. Esses k *bits* são chamados de mensagem e constituem um grupo de 2^k mensagens possíveis. No processo de codificação, o codificador transformará cada bloco de k *bits* em um bloco de n *bits* chamado de palavra-código, com $n > k$. Portanto, cada palavra-código terá $n - k$ *bits* de paridade ou redundância, que serão usados no processo de decodifição e depois serão eliminados.

Os códigos de bloco são usualmente representados pela notação $C(n, k)$ e sua taxa pode ser calculada pela expressão $R = k/n$, que é um parâmetro indicador de seu nível de redundância. Um código com valor de R igual a R_1 ocupará menos memória física no decodificador se comparado com um código cujo valor de R é igual a R_2, supondo-se $R_1 < R_2$.

Outra característica importante dos códigos de bloco é que são códigos sem memória, ou seja, a redundância ou paridade depende apenas da palavra a ser codificada. Alguns exemplos conhecidos desse tipo de código são: códigos de Hamming, Reed-Muller, Golay, BCH e Reed-Solomon.

3.2.1.1 Códigos de Bloco Lineares

Seja um código de bloco binário $C(n, k)$. Se as 2^k palavras-código pertencentes ao código em questão formarem um subespaço vetorial S, de dimensão k, do espaço vetorial V_n, então $C(n, k)$ é classificado como um **código de bloco linear** [22]. Nesse tipo de

código, a soma de duas palavras-código resulta em outra palavra-código.

Existe uma classe importante de códigos de blocos lineares chamada de **códigos cíclicos**, caracterizados pelo seguinte fato: ao rotacionar os bits de qualquer uma de suas palavras-código em i posições para a direita, o vetor resultante também será uma palavra-código válida [23]. Isto é, se os elementos de uma palavra-código $c = (c_0, c_1, ..., c_{n-1})$ pertencente a um código cíclico C forem rotacionados para a direita em uma posição, o vetor resultante $c' = (c_{n-1}, c_0, c_1, ..., c_{n-2})$ também será uma palavra-código válida de C. Essa característica torna relativamente simples a implementação desse tipo de código por meio de registradores de deslocamento.

Os códigos cíclicos podem ser manuseados algebricamente usando-se polinômios definidos sobre $GF(2^m)$. Por exemplo, a palavra-código $c = (c_0, c_1, ..., c_{n-1})$ pertencente a um código C_{cic} pode ser representada sob a forma polinomial mostrada pela Equação (3.12), onde os coeficientes c_i pertencem a $GF(2^m)$ e o expoente i é um número inteiro que corresponde à posição do elemento na palavra-código. As operações de adição e multiplicação entre dois polinômios $c_1(X)$ e $c_2(X)$, definidos sobre o campo binário $GF(2)$, podem ser realizadas fazendo-se as operações de soma e multiplicação módulo-2 (mostradas na Tabela 3.1) entre os coeficientes dos dois polinômios que possuem o mesmo expoente X^i.

$$c(X) = c_0 + c_1(X) + ... + c_{n-1}X^{n-1} \tag{3.12}$$

Um código cíclico $C_{cic}(n, k)$ possui ainda um polinômio de grau mínimo (igual a $n-k$), diferente de zero, chamado de polinômio gerador desse código. Tal polinômio tem a forma mostrada na Equação (3.13) e, como qualquer outro polinômio do código é multiplo de $g(X)$, esse polinômio determina e gera o código $C_{cic}(n, k)$. Isso quer dizer que, de posse do polinômio mensagem $m(X)$, é possível codificá-la (obter a palavra-código) simplesmente fazendo-se a operação $c(X) = m(X)g(X)$.

$$g(X) = 1 + g_1(X) + ... + g_{n-k-1}X^{n-k-1} + X^{n-k} \tag{3.13}$$

Os códigos BCH constituem um importante exemplo de códigos de bloco cíclicos. Os códigos Reed-Solomon também são códigos de bloco cíclicos já que são uma sub-família dos códigos BCH cujos elementos são não binários, ou seja, pertencem ao campo $GF(q)$, onde $q > 2$.

Outro grupo importante de códigos de bloco lineares é formado pelos códigos LDPC (*low-density parity check*). Esse tipo de código foi originalmente introduzido em 1962 por Robert G. Gallager em sua tese de doutorado, mas seu potencial não pôde ser completamente explorado devido à limitada capacidade computacional da época. Posteriormente, na década de 1990, David J. C. MacKay e Radford M. Neal introduziram uma nova classe de códigos de bloco que logo foi reconhecida como sendo uma redescoberta do código LDPC proposto por Gallager.

Os códigos LDPC são muito poderosos, podendo ter desempenho próximo do limite proposto por Shannon. Alguns exemplos de aplicação que podem ser citados são os padrões 10GBase-T Ethernet e Wi-Fi 802.11.

3.2.2 Códigos Convolucionais

São códigos em que há um fluxo contínuo de dados, entrelaçando os símbolos de redundância com os de informação. A codificação se dá da seguinte maneira: um conjunto de m símbolos de entrada são transformados em n símbolos e essa transformação é função dos últimos k símbolos na entrada do codificador. A razão m/n é conhecida por taxa do código.

Como seus símbolos de redundância dependem das últimas palavras codificadas, os códigos convolucionais são códigos com memória. Outra característica desse tipo de código é que o algoritmo de Viterbi é usado frequentemente no processo de decodificação.

3.2.3 Códigos Concatenados

Os **códigos concatenados** formam uma classe de códigos corretores de erro introduzida por Dave Forney na década de 1960, que combina paralela ou serialmente dois ou mais códigos pertencentes às classes citadas nas seções anteriores, conforme ilustrado nas Figuras 3.1 e 3.2. Graças às características individuais de cada código e também à decodificação iterativa, consegue-se obter uma otimização de desempenho.

Figura 3.1: Diagrama de blocos de um código concatenado serial genérico.

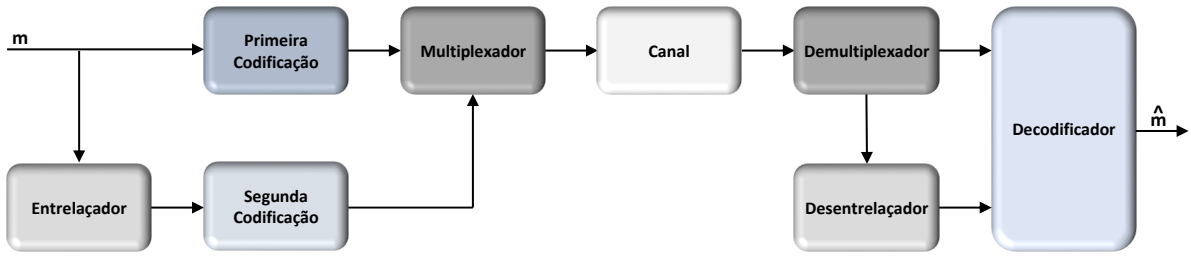

Figura 3.2: Diagrama de blocos de um código concatenado paralelo genérico.

Os códigos concatenados podem receber uma subclassificação, dependendo da classe à qual pertencem os blocos utilizados: bloco-bloco, bloco-convolucional ou convolucional-convolucional. Os códigos que merecem destaque são os chamados códigos turbo, que foram introduzidos por Berrou, Glavieux e Thitimajshima em 1993 quando propuseram um esquema de concatenação paralela de dois códigos convolucionais binários decodificados servindo-se de decodificação interativa.

Uma informação importante que relaciona os códigos concatenados e o presente trabalho é que, dentre os oito códigos corretores de erro do tipo EFEC propostos no anexo da recomendação G.975.1 do ITU-T [3], seis são do tipo concatenado; incluindo o código do anexo I.9, tema deste trabalho.

Capítulo 4

O Protocolo OTN

4.1 Origem

Assim como tudo relacionado a tecnologia, as redes de transporte de dados também evoluíram muito nas últimas décadas. Inicialmente, as informações eram transmitidas de forma analógica através de cabos coaxiais, o que limitava consideravelmente o alcance da transmissão. Nos anos 70 surgiram técnicas de transmissão digital como a de multiplexação por divisão de tempo (TDM, *time division multiplexing*), que possibilitaram um aumento na relação sinal-ruído e, consequentemente, um enlace com alcance mais longo.

Desenvolvida para transportar voz em redes públicas de telefonia comutada (PSTN, *public switched telefone network*), a técnica TDM foi aliada ao protocolo de hierarquia digital plesiócrona (PDH, *plesiochronous digital hierarchy*) cujo nome vem do fato de cada canal multiplexado operar com um relógio não sincronizado com o relógio dos demais canais, apesar de todos possuirem a mesma especificação. As redes que operam com o protocolo PDH são hierarquicamente compostas por níveis (ordens), onde o sinal agregado de cada ordem é formado pela junção de quatro sinais denominados tributários [24].

Devido a fatores como capacidade de gerenciamento e manutenção de rede limitadas e também a dificuldade de interoperabilidade entre redes devido à falta de uma padronização a nível mundial, o PDH foi perdendo espaço para um outro protocolo compatível com ele, porém mais flexível e com maior capacidade de gerenciamento. Padronizado pelo ITU-T nos anos 80, esse novo protocolo é chamado hierarquia digital síncrona ou SDH (*synchronous digital hierarchy*) [25]. Diferentemente do PDH, no SDH os relógios dos tributários são sincronizados a uma única referência de relógio, o que justifica o nome do protocolo.

Os primeiros estudos dessa nova tecnologia de transmissão hierárquica ocorreram nos laboratórios Bellcore (*Bell communications research*) com o protocolo Sonet (*synchronous optical network*), que posteriormente foi padronizado pelo Ansi (*american national standards institute*). Esse protocolo tem características e taxas muito similares ao que foi padronizado pelo ITU-T como o SDH, entretanto é mais comum na América do Norte, enquanto que o SDH é mais encontrado no restante do mundo.

O Sonet e o SDH têm estrutura de multiplexação que permite o transporte de sinais PDH, mas constituem uma evolução muito grande sobre esse protocolo pois, além das vantagens já citadas, elevaram os níveis de multiplexação e também as taxas de transmissão de 2048 Kb/s a 139264 Kb/s, que são respectivamente as taxas dos níveis hierárquicos mais baixo (E1) e mais alto (E4) para o PDH, para 51,84 Mb/s a 39813,12 Mb/s, que são respectivamente as taxas para os níveis hierárquicos mais baixo (STM-0) e mais alto (STM-256) do SDH. Todavia, por terem sido desenvolvidos para transportar sinais de voz em telefonia, não são muito eficientes para transportar dados. Com o crescimento das redes de pacotes, principalmente Ethernet, surgiram novos equipamentos SDH, chamados de SDH-NG (*next generation*), que incluem mecanismos para melhorar a eficiência no transporte de dados [24].

Enquanto as tecnologias de transmissão óptica foram se aprimorando, o tráfego de dados aumentava. Em razão disso, foi preciso ampliar a sinergia entre as redes de dados e as redes ópticas. Estudos iniciados nos anos 90 deram origem a um novo protocolo chamado OTN (*optical transport network*), padronizado pela primeira vez ainda nessa década pelo ITU-T, com a recomendação G.709 [2].

O protocolo OTN possui várias vantagens sobre os padrões anteriores, entre elas a necessidade um menor número de conversões eletro-ópticas, a melhoria nas funções de gerenciamento e supervisão e ainda a inclusão de um campo no quadro dedicado para alocação de dados destinados à correção de erros (FEC). Conforme ilustra a Figura 4.1, diferentes tipos de protocolo incluindo Ethernet, Sonet/SDH e o próprio OTN podem ser encapsulados de forma transparente em uma única estrutura de unidade de transporte óptico denominada OTU (*optical transport unit*), motivo pelo qual esse padrão é conhecido pelo termo em inglês *digital wrapper*, que pode ser traduzido para o português como “envelope digital”.

A continuidade da evolução das redes Ethernet, especialmente com o desenvolvimento

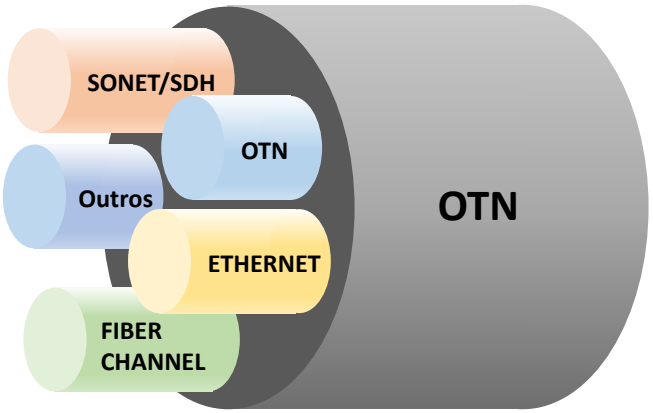

Figura 4.1: Encapsulamento de protocolos.

das tecnologias de 40 e 100 Gigabit Ethernet, levou a um reestudo do padrão OTN. Em dezembro de 2009, foi aprovada uma nova versão da recomendação G.709, na qual novas capacidades e o suporte para transportar sinais Ethernet com altas taxas foram propostas.

Características como a possibilidade de se usar os comprimentos de onda dos sistemas WDM (*wavelenght division multiplexing*) de forma flexível e otimizada, agregando vários tipos de tráfego menores que um comprimento de onda (sub-λs) em um único comprimento de onda fazem do protocolo OTN o padrão atual para transmissão em redes ópticas, sendo capaz de transmitir com taxas superiores a 100 Gb/s de forma eficiente, confiável e robusta.

4.2 Hierarquia

As redes OTN são estruturadas hierarquicamente em três camadas ópticas: OCh (*optical channel*), OMS (*optical multiplex section*) e OTS (*optical transmission section*), conforme ilustrado pela Figura 4.2.

A camada OCh transporta dados digitais de cliente entre pontos da rede onde o sinal sofre procedimentos chamados de regeneração 3R, ou seja: reamplificação, reformatação e retemporização. Suas subcamadas estão no domínio elétrico e são chamadas de: OPUk (*optical payload unit*), ODUk (*optical data unit*) e OTUk (*optical transport unit*), onde k = 1, 2, 3 e 4 indica a taxa de operação. A subcamada OPUk se inicia no ponto em que o sinal de cliente é encapsulado em um sinal OTN e termina onde o sinal é extraído. A subcamada ODUk tem origem e fim nos mesmos pontos em que a OPUk se origina e termina. Já a subcamada OTUk é gerada e terminada em qualquer ponto onde há uma conversão do sinal do meio elétrico para o óptico ou vice-versa. Ao ser convertido para um

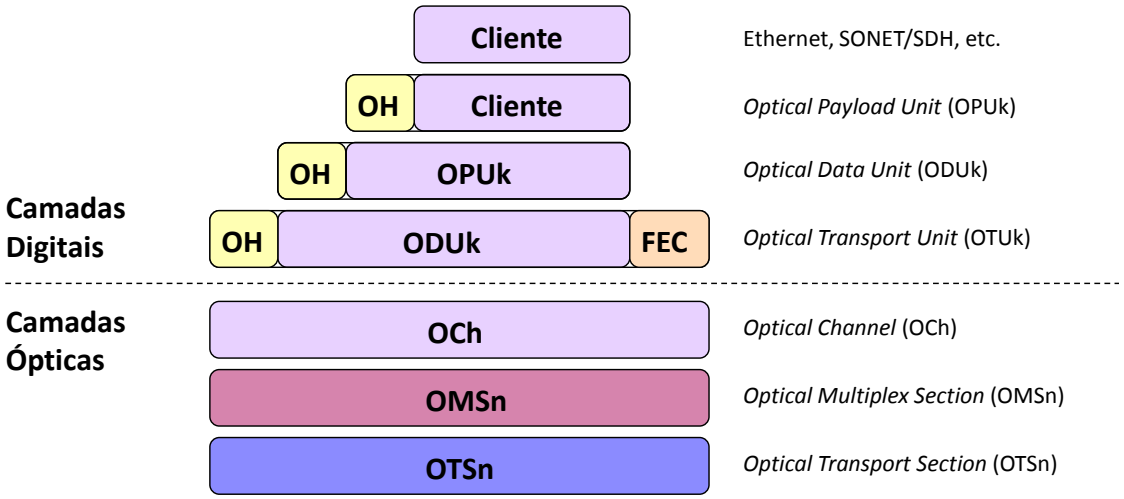

Figura 4.2: Camadas que constituem a hierarquia OTN.

sinal óptico, o sinal OTUk passa a ser um sinal do canal óptico (OCh).

Assim como a OCh, as camadas OMS e OTS estão no domínio óptico. A função da camada OMS é fornecer funcionalidade de rede de um sinal óptico de múltiplos comprimentos de onda (cor), garantindo a integridade dos vários comprimentos de onda das informações da OMS. À camada OTS cabe fornecer funcionalidade para transmissão de sinais ópticos em meios ópticos de diferentes tipos (fibras específicas, por exemplo). Dessa maneira, diversos canais podem ser mapeados dentro da OMS e, então, transportados via camada OTS [24].

4.3 Estrutura de Quadro

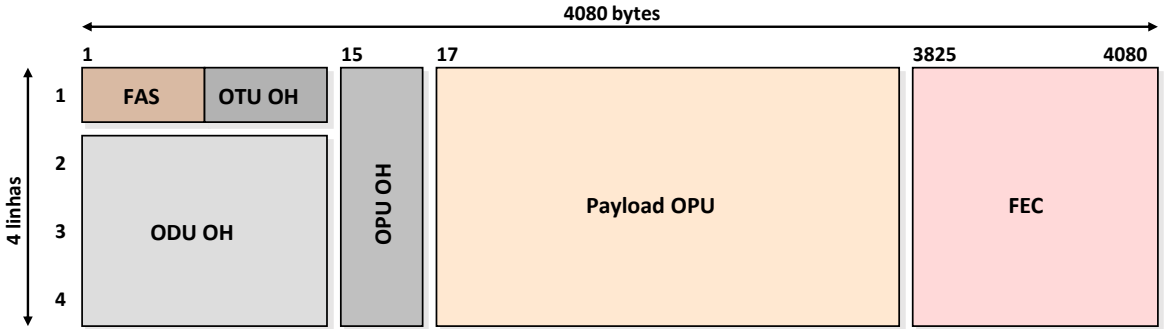

Figura 4.3: Estrutura do quadro usado em redes OTN (quadro OTU).

O quadro usado em redes OTN possui um total de 130560 *bits*, divididos em 4 linhas de 4080 *bytes* ou 32640 *bits* cada. Os *bits* são transmitidos da esquerda para a direita, de

cima para baixo. Na Figura 4.3 é possível notar as seis diferentes áreas, cujas funções são:

- **FAS (*frame alignment signal*):** Campo com tamanho de 6 *bytes* que abriga uma sequência igual a F6F6F6282828 em hexadecimal, utilizada para alinhamento de quadro no receptor. É considerado parte do cabeçalho de OTU (OTU OH).

- **OTU OH:** Trata-se do cabeçalho de OTU. É um campo com 8 *bytes* que abriga informações sobre identificação/estado do quadro e também uma sequência chamada de MFAS (*multiframe alignment signal*), usada para alinhamento de multiquadro (conjunto de 256 quadros). O campo MFAS tem tamanho de um *byte* e é incrementado de uma unidade a cada quadro.

- **ODU OH:** Trata-se do cabeçalho de ODU. É um campo com dimensões iguais a três linhas por quatorze *bytes* e que abriga *bytes* e *bits* para monitoramento da rede. O cabeçalho de ODU possibilita também monitorar erros de transmissão, falhas, defeitos e possui ainda uma área de tamanho igual a dois *bytes* chamada de GCC (*general communications channel*) destinada à comunicação geral entre dois elementos da rede com acesso ao quadro ODUk.

- **OPU OH:** Trata-se do cabeçalho de OPU e tem dimensões iguais a quatro linhas de 2 *bytes*. Abriga informações sobre o tipo de dado que está ocupando o campo OPUk e também *bits* de justificação para compensar diferenças entre taxas de transmissão e o tipo de cliente mapeado.

- ***Payload* OPU:** Campo com dimensões iguais a quatro linhas de 3808 *bytes* destinado à acomodação dos dados de cliente.

- **FEC:** Campo com dimensões iguais a quatro linhas de 256 *bytes* destinado à acomodação dos dados de paridade, usados para detecção e correção de erros no receptor. O código corretor de erros determinado pela recomendação do ITU-T G.709 é o Reed-Solomon é o RS(255,239), no qual cada símbolo é constituído por 8 *bits* ou 1 *byte*. A sequência de *bits* a ser codificada é subdividida em 239 *bytes* e 16 *bytes* de paridade são adicionados para compor a palavra-código de 255 *bytes*. A capacidade de correção desse código é de 8 símbolos, permitindo que o decodificador corrija até 64 *bits* por palavra-código.

4.4 Taxas de Transmissão

As taxas de transmissão dos quadros OTN são padronizadas pela G.709 e são expressadas em função da taxa do padrão SDH STM-16, ou seja, 2048,32 Kb/s.

4.4.1 Quadros OTUk

Os quadros OTUk, para $k = 1, 2, 3, 4$, são formados pela somatória de todas as 6 áreas mostradas na Figura 4.3, totalizando 4 linhas de 4080 bytes. A taxa de *bits* do quadro OTU1 é igual a 255/238 vezes a taxa do STM-16. Para OTU2, tem-se a taxa 255/237 vezes a taxa de 4 vezes STM-16. Em OTU3, a taxa sobe para 16 vezes 255/236 vezes STM-16 e, por fim, em OTU4 a taxa é de 40 vezes 255/227 vezes STM-16. A Tabela 4.1 reúne essas informações adicionando a tolerância atribuída a cada taxa.

Tabela 4.1: Taxa de dados para quadros OTUk.

Tipo	**Taxa Nominal (kb/s)**	**Tolerância**
OTU1	(255/238) · 2488,32	+/-20 ppm
OTU2	(255/237) · 9953,28	+/-20 ppm
OTU3	(255/236) · 39813,12	+/-20 ppm
OTU4	(255/227) · 99532,80	+/-20 ppm

4.4.2 Quadros ODUk

Os quadros ODUk, para $k = 1, 2, 3, 4$, são constituídos pelas seguintes áreas mostradas na Figura 4.3: FAS, OTU OH, ODU OH, OPU OH e Payload OPU, sendo que as áreas FAS e OTU OH são reservadas para alinhamento de quadro e cabeçalho específico OTUk. Com 4 linhas de 3824 *bytes*, um quadro ODUk possui um total de 122368 *bits*.

Assim como nos quadros OTUk, os valores das taxas ODUk são múltiplos da taxa STM-16. A Tabela 4.2 mostra as taxas de *bit* e tolerância para os diferentes tipos de quadro ODU, incluindo o ODU2e que tipicamente é aplicado em transporte de sinais 10 Gigabit Ethernet ou 10 Gigabit Fiber Channel e o ODUflex que pode ser usado para transportar sinais com taxa de fluxo de dados constante (CBR, *constant bit rate*) como 8G Fiber Channel e Infiniband:

Tabela 4.2: Taxa de dados para quadros ODUk.

Tipo	**Taxa Nominal (kb/s)**	**Tolerância**
ODU0	1244,16	+/-20 ppm
ODU1	(239/238) · 2488,32	+/-20 ppm
ODU2	(239/237) · 9953,28	+/-20 ppm
ODU3	(239/236) · 39813,12	+/-20 ppm
ODU4	(239/227) · 99532,80	+/-20 ppm
ODU2e	(239/237) · 10312,50	+/-100 ppm
ODUflex	taxa do cliente	cliente

4.4.3 Quadros OPUk

Os quadros OPUk, para $k = 1, 2, 3, 4$, são constituídos pelas áreas OPU OH e Payload OPU mostradas na Figura 4.3, com um total de 4 linhas de 3808 *bytes* ou 121856 *bits*, dos quais 8 *bytes* são referentes ao cabeçalho OPU e o restante corresponde aos dados de um tributário.

Como mencionado anteriormente, os valores das taxas OPUk são múltiplos da taxa STM-16. A Tabela 4.3 mostra as taxas de *bit* e tolerância para os diferentes tipos de quadro OPU, incluindo o OPU2e, o OPUflex e o OPU0, que é utilizado para mapeamento de quadros 1 Gigabit Ethernet:

Tabela 4.3: Taxa de dados para quadros OPUk.

Tipo	**Taxa Nominal (kb/s)**	**Tolerância**
OPU0	(238/239) · 1244,16	+/-20 ppm
OPU1	2488,32	+/-20 ppm
OPU2	(238/237) · 9953,28	+/-20 ppm
OPU3	(238/236) · 39813,12	+/-20 ppm
OPU4	(238/227) · 99532,80	+/-20 ppm
OPU2e	(238/237) · 10312,50	+/-100 ppm
OPUflex	taxa do cliente	cliente

Capítulo 5

Implementação dos Códigos EFEC (*Enhanced Forward Error Correction*)

5.1 Informações Gerais

Como já foi dito, os dois códigos corretores de erro que são o tema deste trabalho pertencem a um grupo denominado *Enhanced* FEC (EFEC) ou *Super* FEC e são chamados dessa maneira pelo seu desempenho aprimorado em relação ao *Generic* FEC (GFEC), que é o código corretor de erro padrão para redes OTN e cujas características podem ser encontradas no anexo A da recomendação G.709 do ITU-T [2].

O primeiro dos dois códigos, chamado nesta dissertação de **EFEC I.9 original**, é formado por dois BCH(1020,988) entrelaçados, tal como descrito no anexo I.9 da recomendação G.975.1. Já o segundo código, denominado **EFEC I.9 modificado**, foi desenvolvido a partir do primeiro, mas conta com um aumento na quantidade de redundância para melhorar o desempenho. Ambos os códigos foram implementados em FPGA, em liguagem VHDL, para operar a uma taxa de 2,5 Gbps (compatível com OTU1) e têm como cenário de uso um enlace ponto-a-ponto entre equipamentos de transmissão óptica (*transponders*).

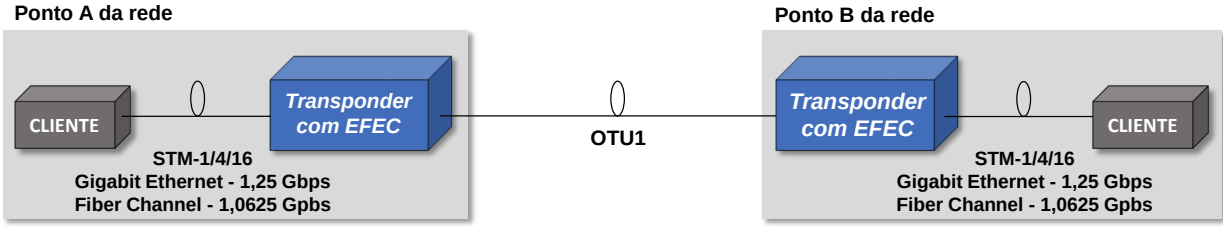

Figura 5.1: Cenário de uso dos EFECs.

5.2 Algoritmo do EFEC I.9 Original

Os dois códigos BCH(1020,988) que compõem o EFEC I.9 possuem comprimentos de bloco iguais, sendo que o tamanho total do bloco de informação é igual a 522240 *bits*, enquanto o tamanho do bloco de informação útil é igual a 489472 *bits*. A taxa de paridade é igual a

$$\frac{n-k}{k} = \frac{522240 - 489472}{489472} = \frac{32768}{489472} \approx 0,06694 \approx 7\%. \tag{5.1}$$

Pode-se dizer, portanto, que o valor da taxa de paridade do EFEC I.9 está perfeitamente alinhado com o valor da taxa de paridade definido na recomendação G.709, cujo cálculo é dado por

$$\frac{n-k}{k} = \frac{130560 - 122368}{122368} = \frac{8192}{122368} \approx 0,06694 \approx 7\%. \tag{5.2}$$

5.2.1 Código BCH(1020,988)

O BCH(1020,988) é um código de bloco cíclico binário, cujas palavras-código têm mil e vinte *bits* de comprimento, dos quais trinta e dois são paridade. Ele é derivado do BCH(1023,993), que tem parâmetros $t = 3$ e $m = 10$, o que significa que até três erros podem ser corrigidos por palavra-código e trinta *bits* de paridade são requeridos para tal. No BCH(1020,988) a área de dados é cinco *bits* menor do que no código BCH(1023,993) e à sua área de paridade são acrescidos dois *bits* para proteção adicional. O polinômio de campo é

$$p(x) = x^{10} + x^3 + 1. \tag{5.3}$$

Os códigos BCH(1020,988) entrelaçados com base nos quais são formados os EFECs deste trabalho são diferentes entre si. Portanto, há um polinômio gerador para cada um desses códigos: um deles gera as palavras-código da codificação chamada de ***horizontal*** e o outro as palavras-código da codificação chamada de ***slope*** (inclinado). Tais nomes são atribuídos devido a características do remapeamento dos dados que ocorre nos processos de codificação, assuntos que serão abordados nas próximas seções deste documento. Os polinômios geradores dos códigos *horizontal* e *slope* são

$$g_H(x) = m_1(x)m_3(x)m_5(x)(x^2 + 1) \tag{5.4}$$

$$g_S(x) = x^{30}m_1(x^{-1})m_3(x^{-1})m_5(x^{-1})(x^2 + x + 1). \tag{5.5}$$

Os termos (x^2+1) e (x^2+x+1) trazem proteção adicional contra detecção falsa de erro e os termos m_1, m_3 e m_5 são polinômios mínimos padrões do código BCH(1023,993), definidos por

$$m_1(x) = x^{10} + x^3 + 1 \tag{5.6a}$$

$$m_3(x) = x^{10} + x^3 + x^2 + x + 1 \tag{5.6b}$$

$$m_5(x) = x^{10} + x^8 + x^3 + x^2 + 1. \tag{5.6c}$$

5.2.2 Estrutura de Quadro

Os processos de codificação *horizontal* e *slope* são realizados sobre estruturas matriciais de dados, específicas para cada caso, denominadas **superquadros**. Em se tratando do EFEC I.9 original, tanto o superquadro *horizontal* quanto o *slope* têm dimensões iguais a 512 linhas por 1024 colunas, conforme ilustra a Figura 5.2. Cada coordenada linha *versus* coluna é ocupada por um único *bit* e o conjunto de *bits* que pertencem a uma linha do superquadro constituem uma palavra-código.

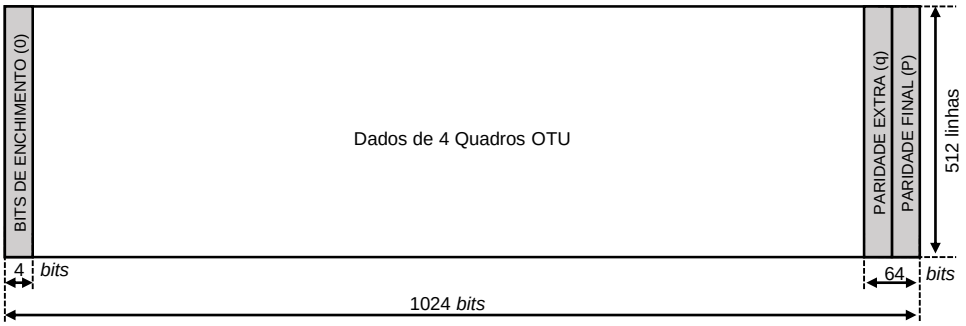

Figura 5.2: Formatos dos superquadros do EFEC I.9 original.

É possível notar quatro áreas demarcadas no superquadro da Figura 5.2. A área maior, em branco, tem dimensões iguais a 512 linhas por 956 colunas é constituída pelos dados de 4 quadros ODU, adicionados dos respectivos cabeçalhos OTU, o que corresponde a 489472 *bits*. A área denominada paridade extra (q) tem dimensões iguais a 512 linhas por 32 colunas e se refere a dados de paridade calculados em uma das etapas do cálculo da paridade final (P). Mais detalhes sobre a paridade extra serão fornecidos mais adiante neste documento. A área localizada mais à direita abriga a paridade final, que tem as mesmas dimensões da paridade extra e corresponde à paridade do código BCH(1020,988). Finalmente, a área mais à esquerda tem dimensões iguais 512 linhas por 4 colunas e contém

bits de enchimento iguais a zero.

Cada linha de um superquadro contém uma palavra-código BCH(1020,988) adicionada dos *bits* de enchimento nas posições de 0 a 3. Como serão processados primeiro, os *bits* de enchimento não interferem no resultado final da codificação e só são necessários para que seja possível subdividir os superquadros em sub-blocos para realizar o algoritmo de entrelaçamento exposto na próxima seção. Os *bits* de enchimento obviamente não são transmitidos.

5.2.3 Entrelaçamento

O entrelaçamento no EFEC I.9 é necessário para organizar os dados em superquadros formando, assim, as palavras-código. As características do processo de entrelaçamento fazem com que os dados sejam alocados nos superquadros de maneira distribuída, o que acaba por fornecer proteção contra surtos ou rajadas de erros que podem ocorrer durante a transmissão. Se um surto de tamanho menor ou igual a 1536 erros atingir os dados que formam um par de superquadros *horizontal* e *slope*, o decodificador será capaz de localizar e corrigir todos os *bits* afetados. Apoiado no fato de que os dados de uma linha do quadro ODU são distribuídos verticalmente nos superquadros, o valor da capacidade de correção de surtos mencionado anteriormente é obtido pela multiplicação dos valores que representam a dimensão vertical dos superquadros (512 *bits*) e a capacidade de correção do código BCH(1020,988) que é igual a 3 erros por palavra-código.

Os superquadros são subdivididos em 512 sub-blocos de 32 linhas por 32 colunas, conforme ilustra a Figura 5.3. Ditos sub-blocos são identificados por números para facilitar o entendimento do processo de entrelaçamento dos dados. No caso do superquadro *horizontal*, pode-se ver na Figura 5.3 que o sub-bloco no canto superior esquerdo é identificado pelos números entre parênteses (0,31), o que significa que aquele é o sub-bloco 0 do grupo de coluna 31. As coordenadas do superquadro *horizontal* são tomadas como base nesse processo. Nota-se também que os dois grupos de coluna mais à direita contêm os *bits* 960 a 1023 e abrigam as paridades extra (q) e final (P), respectivamente.

Para formar o superquadro *slope* existe uma fórmula de entrelaçamento, através da qual o *bit* que ocupa a posição [I,J] no superquadro *horizontal* (com $0 \leq \mathrm{I} \leq 511$, $0 \leq \mathrm{J} \leq 1023$) é mapeado na posição [((I-J-1 mod 32) + 32 (I/32 - J/64) mod 512),J] do superquadro *slope*. Examinando-se essa fórmula, nota-se uma importante propriedade do EFEC I.9: a

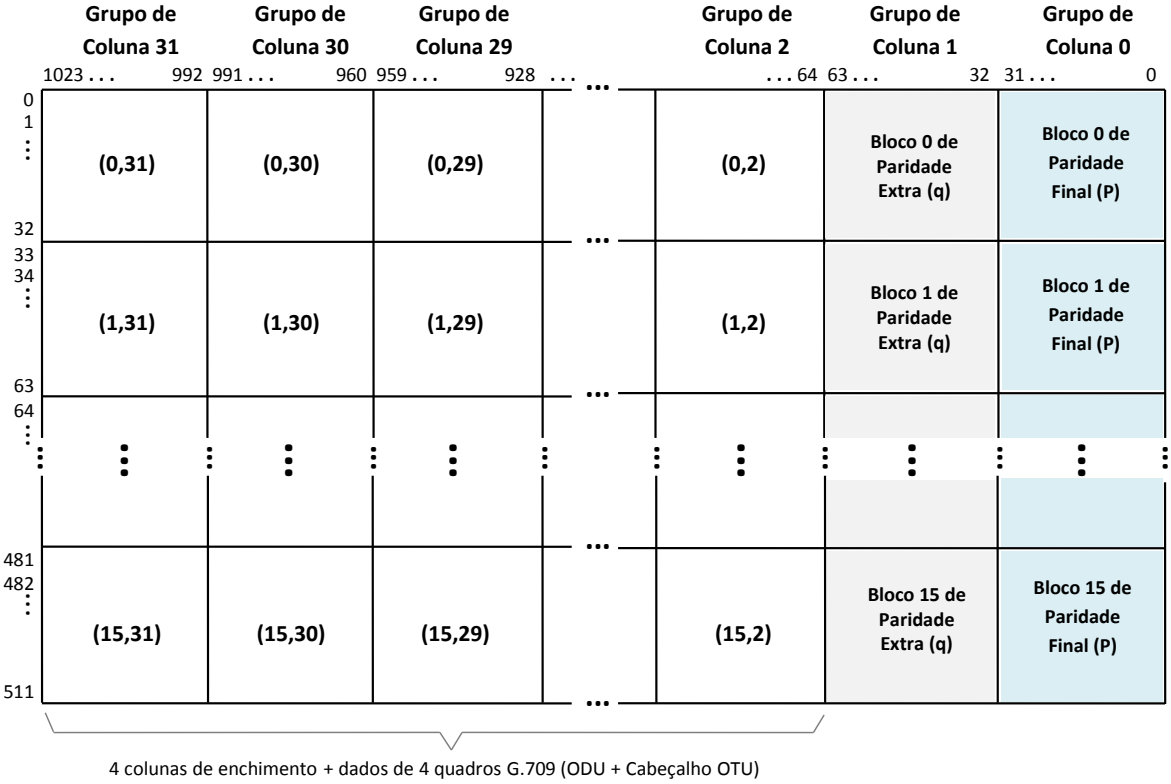

Figura 5.3: Superquadro *horizontal* do EFEC I.9 original dividido em sub-blocos.

coordenada de coluna de um *bit* é sempre a mesma para os dois superquadros. A Figura 5.4 mostra como fica a posição dos sub-blocos após aplicada a fórmula de entrelaçamento.

Para entendimento do processo de entrelaçamento observe, como exemplo, os grupos de coluna 30 e 31. Comparando-se as Figuras 5.3 e 5.4 é possível perceber que os *bits* que estão no sub-bloco 0 do superquadro *horizontal* (coordenadas (i,31) e (i,30), com i variando de 0 a 15) foram alocados nas posições de coordenadas (i+1,31) e (i+1,30) do superquadro *slope*, respectivamente. Isso significa que, para o caso desses dois grupos de coluna, todos os sub-blocos do superquadro *slope* ficam deslocados de uma posição para baixo em relação aos sub-blocos do superquadro *horizontal* e os *bits* do sub-bloco 15 do superquadro *horizontal* são mapeados para os sub-blocos de coordenada (0,31) e (0,30) no superquadro *slope*. Nos grupos de coluna 28 e 29 o processo é o mesmo, porém os sub-blocos do superquadro *slope* ficam deslocados em duas posições em relação ao superquadro *horizontal* ao invés de apenas uma. O processo segue assim até os grupos de coluna 2 e 3, incrementando-se uma posição de diferença entre as posições dos sub-blocos dos dois superquadros a cada par de grupos de coluna. Essa característica dá nome ao superquadro e à codificação *slope*, pois a posição dos *bits* desse superquadro é inclinada (tradução para o português do termo em

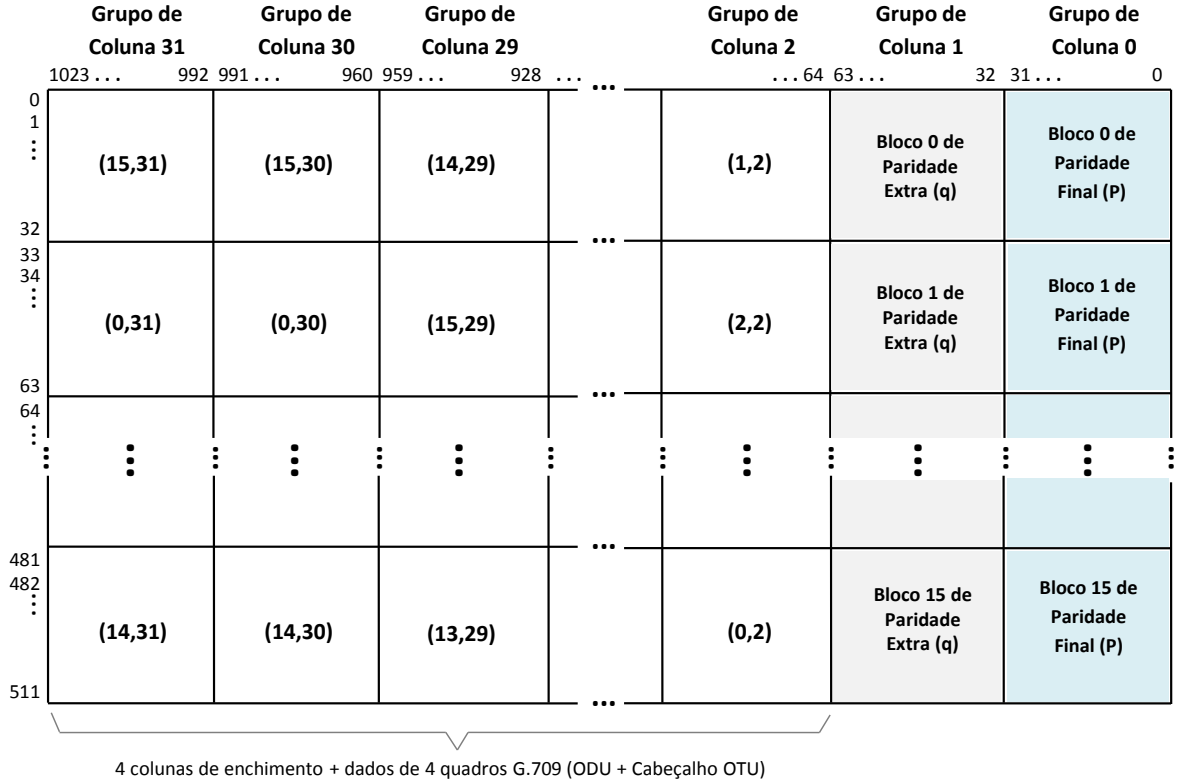

Figura 5.4: Superquadro *slope* do EFEC I.9 original dividido em sub-blocos.

inglês *slope*) em relação à posição dos *bits* do superquadro *horizontal*.

Outra característica do processo de entrelaçamento é que os grupos de coluna que abrigam as paridades extra e final não sofrem entrelaçamento, portanto o conteúdo é o mesmo tanto para o superquadro *horizontal* quanto para o *slope*.

A fórmula de entrelaçamento executa, além do entreçamento de sub-blocos, também o entrelaçamento dos *bits*, alterando a coordenada de coluna dos mesmos dentro de cada sub-bloco *slope*. A Figura 5.5 ilustra tal processo, atribuindo coordenadas à posição de cada *bit* em um sub-bloco *slope* antes e depois do entrelaçamento.

Posição dos *Bits* Antes do Entrelaçamento

0,31	0,30	0,29	···	0,2	0,1	0,0
1,31	1,30	1,29	···	1,2	1,1	1,0
2,31	2,30	2,29	···	2,2	2,1	2,0
3,31	3,30	3,29	···	3,2	3,1	3,0
4,31	4,30	4,29	···	4,2	4,1	4,0
5,31	5,30	5,29	···	5,2	5,1	5,0
6,31	6,30	6,29	···	6,2	6,1	6,0
⋮	⋮	⋮	···	⋮	⋮	⋮
31,31	31,30	31,29	···	31,2	31,1	31,0

Posição dos *Bits* Após o Entrelaçamento

0,31	31,30	31,29	···	3,2	2,1	1,0
1,31	0,30	30,29	···	4,2	3,1	2,0
2,31	1,30	0,29	···	5,2	4,1	3,0
3,31	2,30	1,29	···	6,2	5,1	4,0
4,31	3,30	2,29	···	7,2	6,1	5,0
5,31	4,30	3,29	···	8,2	7,1	6,0
6,31	5,30	4,29	···	9,2	8,1	7,0
⋮	⋮	⋮	···	⋮	⋮	⋮
31,31	30,30	29,29	···	2,2	1,1	0,0

Figura 5.5: Entrelaçamento de *bits* em um sub-bloco do superquadro *slope*.

5.2.4 Codificação

O processo de codificação é sempre realizado sobre o volume de dados que forma um par de superquadros *horizontal* e *slope*. Nesse processo são realizados os cálculos das paridades extra e final, de modo que os dois grupos de coluna mais à direita dos superquadros *horizontal* e *slope* sejam preenchidos e cada linha dos dois superquadros represente uma palavra-código.

Os vetores q, como também é chamada a paridade extra, são um conjunto de 512 vetores de 32 *bits* (um para cada palavra-código) que são calculados para preencher as colunas 63 a 32 (grupo de coluna 1) de cada par de superquadros *horizontal* e *slope*, como se fizessem parte da informação. Esse conteúdo comum aos dois superquadros possibilita que a mesma paridade final (grupo de coluna 0) também possa ser calculada para servir a ambos, tornando necessária a transmissão de apenas 512 vetores de q e 512 vetores de P a cada par formado por um superquadro *horizontal* e um *slope* (ao invés de 512 vetores de q e 512 vetores de P para cada superquadro *horizontal* mais 512 vetores de q e 512 vetores de P para cada superquadro *slope*) e viabilizando a decodificação referente às duas codificações no receptor com esse volume reduzido de dados.

O cálculo dos vetores q é feito em duas etapas. A primeira delas consiste em calcular as chamadas paridades intermediárias dos códigos *horizontal* e *slope*: p_H e p_S, cada qual com um volume de dados igual a 512 vetores de 32 *bits*. Na Figura 5.6, é possível ver uma ilustração do cálculo das paridades intermediárias, onde são mostrados dados de uma palavra-código das duas codificações e também os operadores lineares M_H e M_S, que representam as operações de codificação *horizontal* e *slope*.

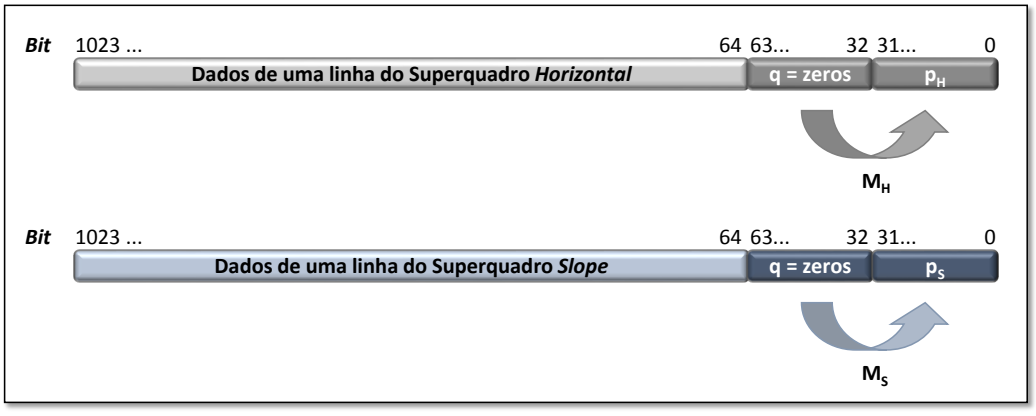

Figura 5.6: Cálculo das paridades intermediárias p_H e p_S.

As paridades p_H e p_S são respectivamente iguais ao resto da divisão entre $r_H(x)$ e $r_S(x)$ (representações polinomiais dos *bits* das colunas 1023 a 32 de uma linha do superquadro *horizontal* ou *slope*) e $g_H(x)$ ou $g_S(x)$, que são os polinômios geradores dos códigos, mostrados nas Equações (5.4) e (5.5). Assim

$$p_H(x) = r_H(x) \bmod g_H(x) \tag{5.7}$$

$$p_S(x) = r_S(x) \bmod g_S(x). \tag{5.8}$$

A implementação das referidas divisões polinomiais foi efetuada empregando-se registradores de deslocamento com conexões de realimentação baseadas nos polinômios geradores $g_H(x)$ e $g_S(x)$ [21].

Como exemplo, suponha o cálculo da paridade intermediária p_H da palavra-código 0 da codificação *horizontal*. Para obter o valor de p_H em questão, deve-se dividir o polinômio que representa as os valores dos *bits* das colunas 1023 a 32 da linha 0 do superquadro *horizontal* pelo polinômio $g_H(x)$. O resto dessa operação será o valor desejado. As colunas 63 a 32, que correspondem ao grupo de coluna 1, sempre devem ter seu conteúdo zerado tanto para os cálculos de p_H como de p_S.

Após efetuar o cálculo dos valores de p_H e p_S, executa-se a segunda e última etapa do cálculo da paridade extra. Para obter a equação de cálculo dessa etapa, pode-se seguir o raciocínio: uma vez que a codificação é uma operação linear, é possível expressar a paridade final como a soma entre a codificação com o vetor q zerado (p_H) e a codificação do vetor q isolada, tal que

$$P = M_H.q + p_H. \tag{5.9}$$

De forma similar, para a codificação *slope*

$$T.P = M_S.T.q + p_S, \tag{5.10}$$

onde T é um operador que representa a operação de entrelaçamento de *bit*.

A Figura 5.7 ilustra o cálculo da paridade final. Sugere-se a comparação com a Figura 5.6 para complementar o entendimento do processo de codificação.

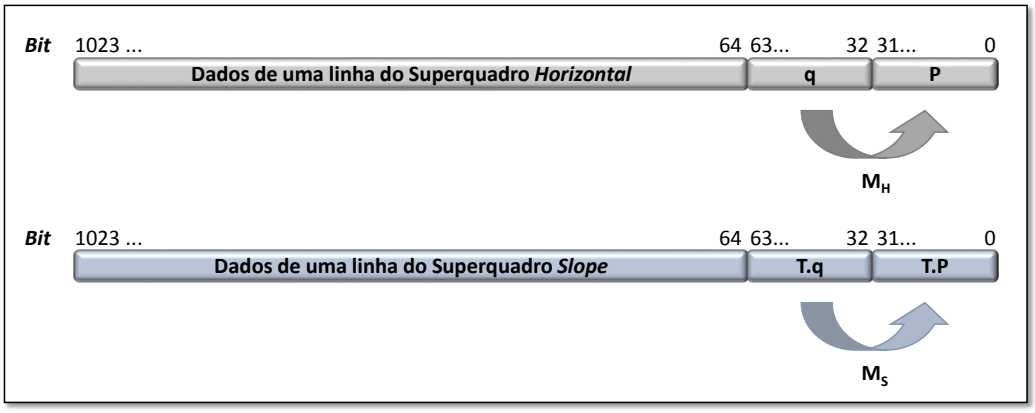

Figura 5.7: Cálculo da paridade final P.

Substituindo-se (5.9) em (5.10), obtém-se

$$T(M_H.q + p_H) = M_S.T.q + p_S. \tag{5.11}$$

Finalmente, isolando-se a variável q na Equação (5.11), obtém-se

$$q = (T.M_H - M_S.T)^{-1}(p_S - T.p_H). \tag{5.12}$$

A Equação (5.12) é usada para o cálculo dos vetores q e alguns de seus termos podem ser interpretados da seguinte maneira:

- $T.p_H$: Dados de p_H com entrelaçamento de *bit*.
- $T.M_H$: Operação de codificação *horizontal*, com entrelaçamento do resultado.
- $M_S.T$: Operação de codificação *slope*, com entrelaçamento antes do cálculo.

De posse dos valores da paridade extra, o próximo passo é calcular a paridade final por meio da Equação (5.9).

5.3 Algoritmo do EFEC I.9 Modificado

No EFEC I.9 modificado também são utilizados dois códigos BCH(1020,988), com os mesmos polinômios geradores do EFEC I.9 original. No entanto, no superquadro do EFEC I.9 modificado o bloco de paridade teve seu tamanho triplicado em relação ao superquadro do EFEC I.9 original, passando de 32768 para 98304 *bits*. Isso fez o número total de *bits*

do superquadro passar de 522240 para 587776 *bits* e a taxa de paridade aumentar de 7% para 20% aproximadamente, conforme o cálculo

$$\frac{n-k}{k} = \frac{587776 - 489472}{489472} = \frac{98304}{489472} \approx 0,20083 \approx 20\%. \tag{5.13}$$

Como consequência do aumento da quantidade de paridade usada no EFEC I.9 modificado, o quadro OTU fica estendido e passa a ter 4 linhas de 4592 *bytes* ao invés de 4 linhas de 4080 *bytes*, como era originalmente. A Figura 5.8 mostra o quadro OTU estendido.

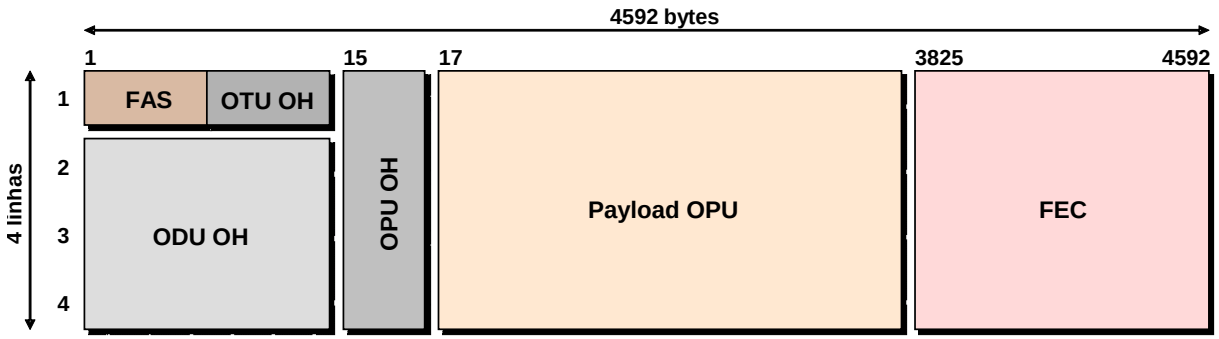

Figura 5.8: Quadro OTU estendido.

5.3.1 Estrutura de Quadro

Esta seção apresenta uma das principais alterações feitas no EFEC I.9 original para dar origem ao EFEC I.9 modificado. Apesar do fato de os dois EFECs compartilharem as codificações *horizontal* e *slope* e também de seus superquadros terem as mesmas dimensões, a maneira como esses superquadros são formados nos dois EFECs e o número total deles é diferente. Enquanto no EFEC I.9 original os dados de quatro quadros ODU eram usados para formar um superquadro *horizontal* e um *slope*, no EFEC I.9 modificado os dados desses quatro quadros ODU são divididos em três partes iguais e distribuídos em três superquadros *horizontal* e três *slope*. A Figura 5.9 compara, para uma das codificações (*horizontal* ou *slope*), como os dados que formaram um superquadro EFEC I.9 original são distribuídos nos três superquadros do EFEC I.9 modificado correspondentes.

5.3.2 Entrelaçamento

O entrelaçamento do EFEC I.9 modificado é feito seguindo as mesmas regras do EFEC I.9 original, ou seja, segundo a Seção I.9.2.3 da recomendação G.975.1 e conforme já foi

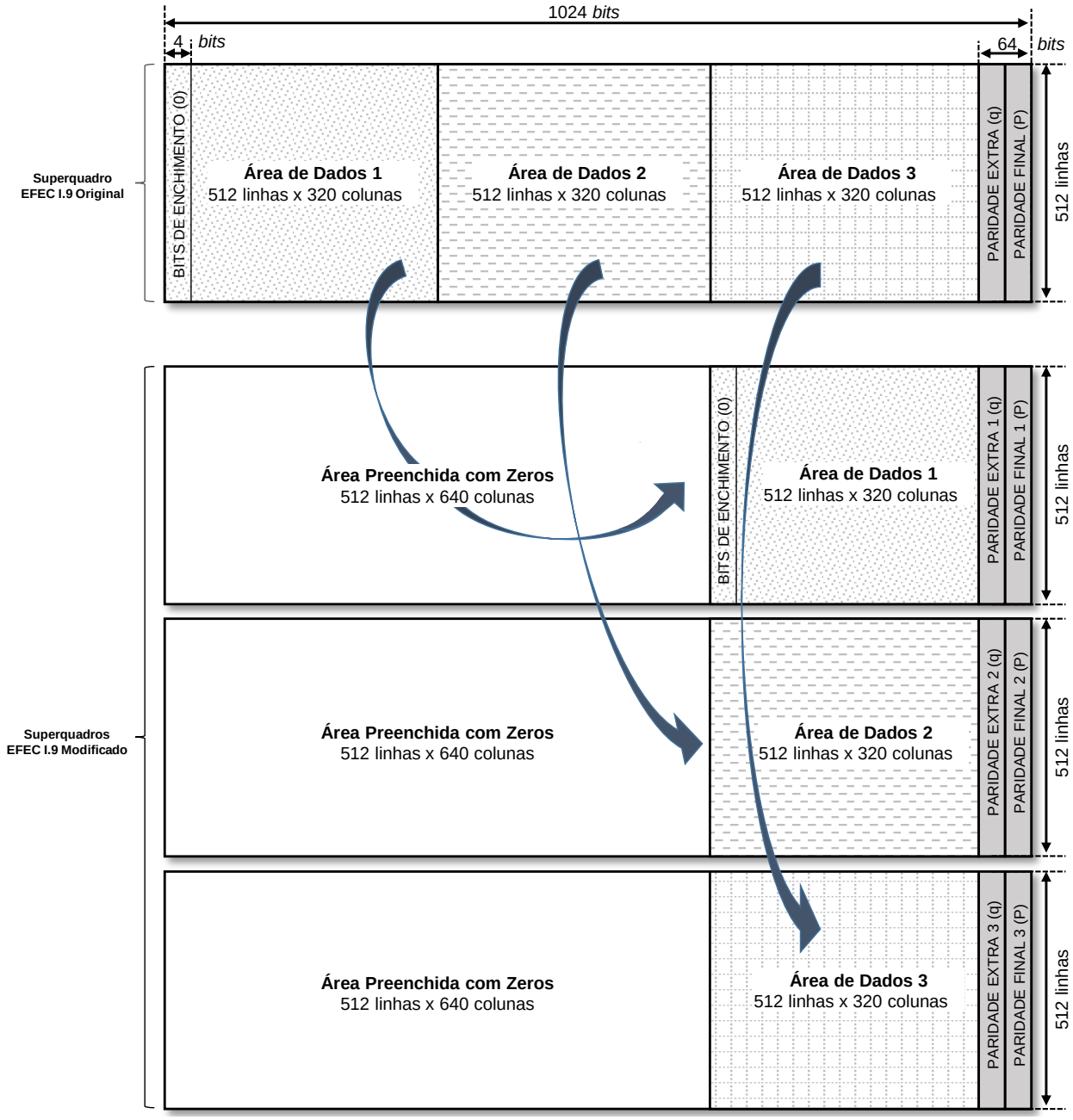

Figura 5.9: Superquadros de uma das codificações dos EFECs original e modificado.

explicado nas seções anteriores deste documento. Os dados resultantes dessa operação serão distribuídos em três superquadros *slope*. A Figura 5.10 mostra o conteúdo dos superquadros de ambas as codificações. Os campos com coordenadas separadas por vírgulas são os sub-blocos.

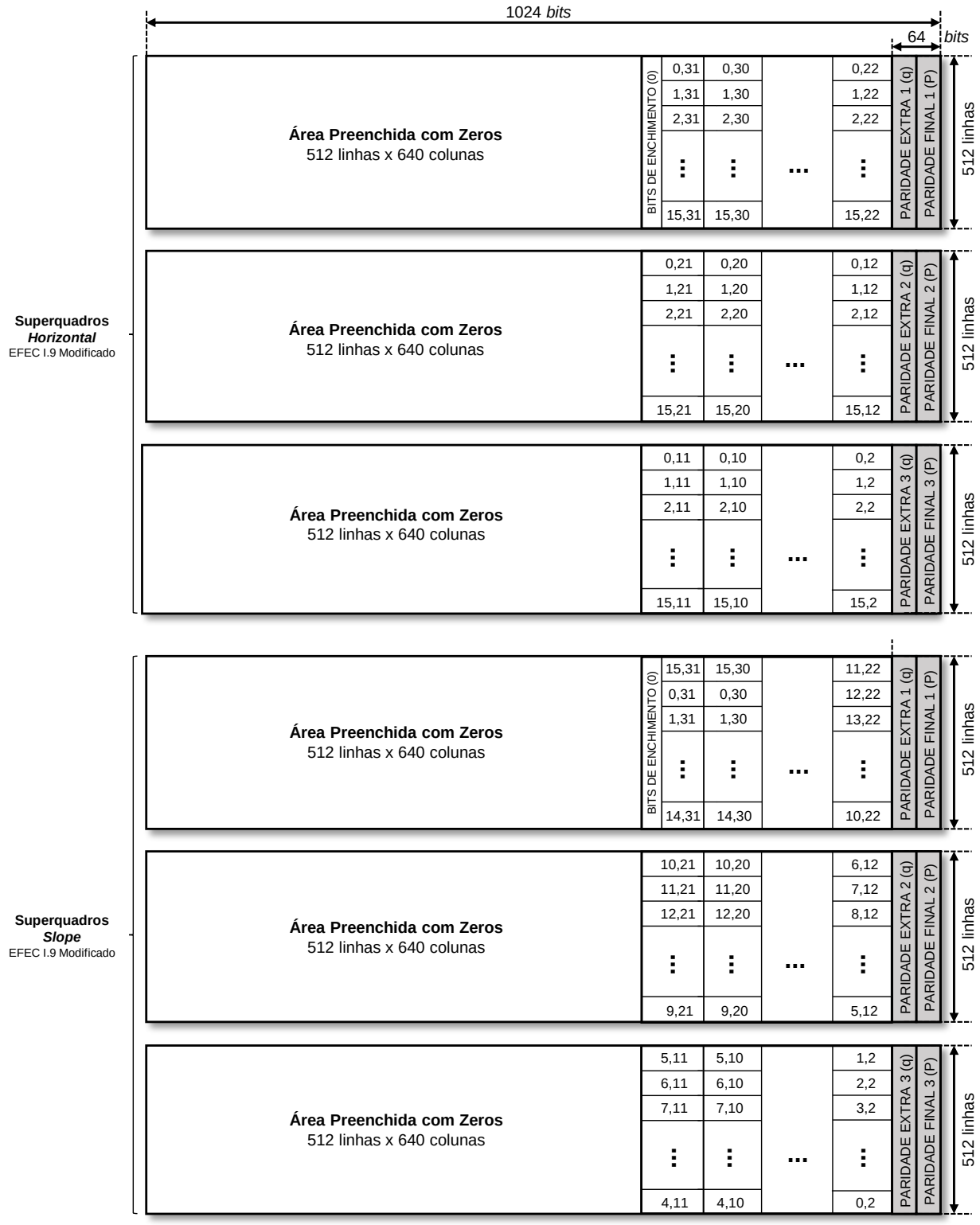

Figura 5.10: Superquadros do EFEC modificado.

5.3.3 Codificação

A exemplo do entrelaçamento, a codificação do EFEC I.9 modificado também segue as mesmas regras do EFEC I.9 original mas, nesse caso, os procedimentos de cálculo das

paridades extra e final têm que ser realizados nos três superquadros de cada codificação mostrados na Figura 5.10, de modo que seus grupos de coluna 0 e 1 sejam preenchidos. Portanto, no EFEC I.9 modificado é calculado três vezes mais paridade que no EFEC I.9 original.

A capacidade de correção do BCH(1020,988) é de até três erros (três *bits*) por palavra-código. As paridades calculadas para os superquadros 1, 2 e 3 da codificação *horizontal* do EFEC I.9 modificado possibilitarão a correção de até três *bits* por palavra-código nos *bits* dos seus grupos de coluna 11 a 2. Tal estratégia confere maior proteção a cada *bit* em comparação com o EFEC I.9 original, refletindo em um maior ganho de codificação. O mesmo raciocínio se aplica aos superquadros *slope*.

5.4 Arquiteturas dos EFECs I.9 Original e Modificado

Nesta seção serão mostrados dados das arquiteturas de projeto usadas para implementar os dois EFECs que são tema deste trabalho. Uma arquitetura de alto nível foi desenvolvida para atender o EFEC I.9 original e a arquitetura do EFEC I.9 modificado partiu desta, fazendo com que os dois projetos tenham várias características em comum. Tais características serão salientadas, bem como os pontos nos quais as arquiteturas divergem.

Uma premissa que norteou este projeto foi, na medida do possível, desenvolver e implementar as operações demandadas pela codificação e decodificação em sub-blocos modulares, permitindo que o número de instâncias em paralelo dessas estruturas seja alterado para expandir a capacidade de processamento. Essa prática visa maximizar a possibilidade de reuso e também da evolução destes EFECs para atender redes OTN que operam em taxas mais elevadas como, por exemplo, 10 Gbps e 100 Gbps.

5.4.1 Codificador

As funções preponderantes do codificador são calcular a paridade sobre os dados de cliente e organizar os dados para transmissão no formato de quadro OTU. A latência decorrente do processo de codificação é da ordem de um superquadro.

A Figura 5.11 é um diagrama de blocos que mostra a arquitetura de alto nível usada para implementação dos codificadores EFEC I.9 original e modificado.

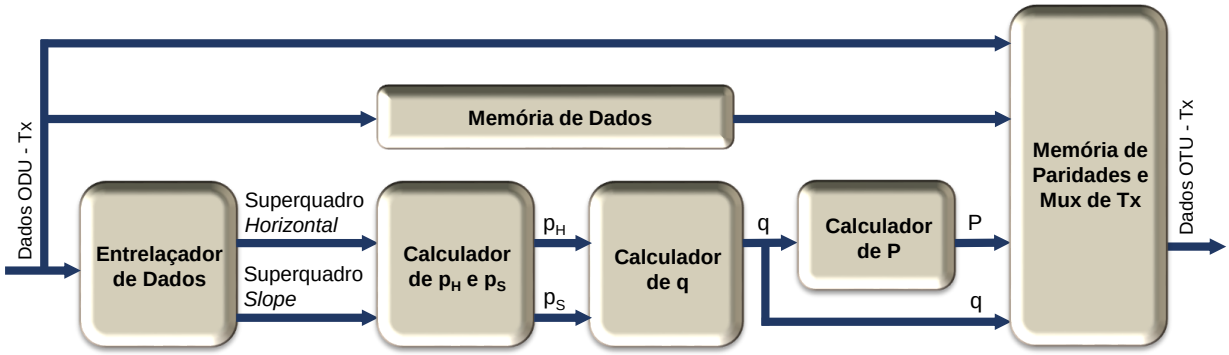

Figura 5.11: Diagrama de blocos dos codificadores EFEC I.9 original e modificado.

5.4.1.1 Memória de Dados

Os dados ODU chegam ao codificador em um barramento paralelo de 32 *bits* de largura e seguem três caminhos: os blocos memória de dados, entrelaçador e memória de paridade e multiplexador de transmissão. Quando o codificador está habilitado, a memória de dados armazena os dados ODU enquanto o processamento das paridades é executado.

Assim como em outros blocos do codificador, no bloco memória de dados é usada a estratégia de paginação de memória. No início da operação os dados de 32 *bits* que chegam ao codificador vão sendo escritos na primeira página de uma memória do tipo RAM (*random-access memory*) até o fim do processamento da paridade do primeiro superquadro. Quando esse processamento termina, a primeira página começa a ser lida para transmissão e os novos dados que chegam são escritos na segunda página. O processo segue invertendo as páginas de escrita e leitura indefinidamente.

5.4.1.2 Entrelaçador

O entrelaçador é o bloco responsável por receber os dados no formato de quadro ODU e entregá-los ao calculador de p_H e p_S, no formato de superquadro, para que as paridades intermediárias sejam calculadas. Tal tarefa é implementada com auxílio de máquinas de estados, memórias e multiplexadores.

São utilizadas 32 instâncias de memória RAM com 2 páginas de 16 endereços de 32 *bits* em cada memória, totalizando 32 *kilobits*. Essa quantidade de memória é suficiente para armazenar o conteúdo de um grupo de coluna completo por página. Cada memória guarda o conteúdo de uma coluna por página, sendo que cada endereço armazena o conteúdo de um sub-bloco. A Figura 5.12 ilustra essa afirmação, mostrando as memórias inseridas no

superquadro e também a ordem com que são preenchidas.

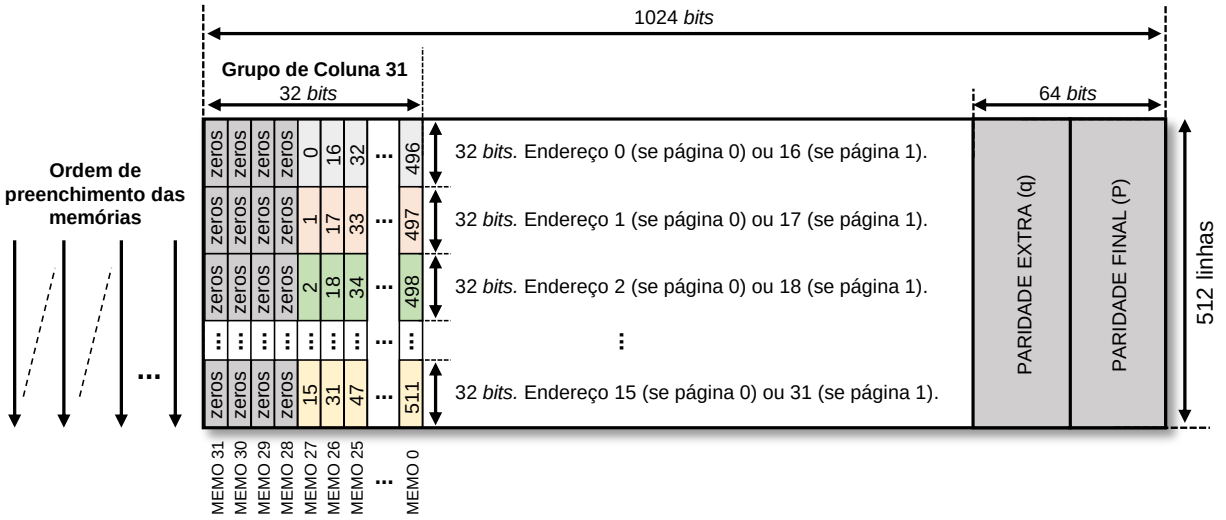

Figura 5.12: Preenchimento das memórias do bloco entrelaçador.

No primeiro ciclo de cada superquadro, o que corresponde ao preenchimento do grupo de coluna 31, a máquina de estados de escrita controla o preenchimento das memórias 28 a 31 com os *bits* de enchimento (zeros). Paralelamente, o fluxo de dados ODU chega em vetores de 32 *bits* que são acomodados, coluna por coluna (memória por memória), em ordem crescente de endereços a partir da memória 27 até a memória 0, conforme ilustrado pelos números dentro de cada posição de endereço na Figura 5.12. Portanto, as memórias sempre são preenchidas no formato correspondente a um grupo de coluna do superquadro *horizontal*.

Quando o grupo de coluna 31 acaba de ser preenchido, a máquina de estados de escrita muda o endereçamento e os dados ODU começam a ser escritos na página 1. Ao mesmo tempo, a máquina de estados de leitura começa a endereçar as memórias buscando os dados que foram escritos na página 0. As páginas das memórias vão sendo escritas e lidas conforme o grupo de coluna e o processo segue assim, indefinidamente.

É possível perceber que, selecionando para leitura um mesmo endereço de todas as memórias, consegue-se ter acesso a quaisquer *bits* do sub-bloco correspondente. Existem dois multiplexadores, um para as palavras-código *horizontal* e outro para as *slope*, que coletam os dados lidos da memória e se encarregam de selecionar os *bits* corretos para serem encaminhados ao calculador de p_H e p_S. Cada multiplexador têm em suas entradas 32 barramentos de 32 *bits* organizados de maneira que se possa ter em sua saída um vetor de 32 *bits* que faça parte de uma palavra-código da codificação à qual ele serve.

O multiplexador de *horizontal* pode levar à saída do entrelaçador qualquer uma de suas 32 entradas, que são (notação: identificação da memória seguido do número do *bit* entre parênteses):

$$\begin{aligned} 0: &\quad M_{31}(0), M_{30}(0), M_{29}(0), ..., M_0(0) \\ 1: &\quad M_{31}(1), M_{30}(1), M_{29}(1), ..., M_0(1) \\ &\quad \vdots \\ 31: &\quad M_{31}(31), M_{30}(31), M_{29}(31), ..., M_0(31) \end{aligned}$$

Como exemplo, considere que o endereço de leitura de todas as memórias está igual ao endereço 0 e que elas estão preenchidas com o conteúdo do grupo de coluna 31. Nesse caso, o multiplexador de *horizontal* consegue levar à sua saída os *bits* 1023 a 992 de qualquer uma das palavras-código das linhas 0 a 15 do superquadro *horizontal*.

Similarmente, o multiplexador de *slope* tem como entradas os vetores de 32 formados da seguinte forma:

$$\begin{aligned} 0: &\quad M_{31}(0), M_{30}(1), M_{29}(2), ..., M_0(31) \\ 1: &\quad M_{31}(1), M_{30}(2), M_{29}(3), ..., M_0(0) \\ &\quad \vdots \\ 31: &\quad M_{31}(31), M_{30}(0), M_{29}(1), ..., M_0(30) \end{aligned}$$

As características de arquitetura do entrelaçador supracitadas estão reunidas no pedido de patente Arquitetura de Circuito e Método para Entrelaçamento de *Bits* em Fluxo de Dados Paralelo, depositado no Instituto Nacional de Propriedade Industrial (INPI) [6].

5.4.1.3 Calculador de Paridades Intermediárias (p_H e p_S)

A função do calculador de paridades intermediárias é utilizar as Equações (5.7) e (5.8) mostradas na Seção 5.2.4 para calcular p_H e p_S. Conforme está escrito na Seção 5.2.4, o procedimento de cálculo envolve dividir o polinômio que representa os *bits* das colunas 1023 a 32 de cada linha dos superquadros (de cada palavra-código) pelo polinômio gerador correspondente. No entanto, para implementar esse procedimento em *hardware* não é interessante executar a divisão dos *bits* 1023 a 32 de cada palavra-código sem interrupção,

pois isso exigiria o armazenamento dos dados de mais de um superquadro por parte do entrelaçador, o que por sua vez se reflete em um consumo desnecessário de área e potência por causa da quantidade de memória que seria usada. Além disso, haveria a necessidade de se esperar o preenchimento praticamente completo dos dados de um superquadro no entrelaçador para iniciar o processamento no calculador de p_H e p_S, o que se traduz em uma latência alta do codificador. Por essas razões, o processamento de p_H e p_S é feito por grupo de coluna, iniciando no grupo 31 e finalizando no grupo 1 (preenchido com zeros) em cada superquadro.

Os dados das palavras-código *horizontal* e *slope* são disponibilizados pelo entrelaçador em dois barramentos dedicados de 32 *bits* de largura. São processados 32 *bits* de uma palavra-código das duas codificações a cada ciclo de relógio, começando na linha 0 e terminando na 511, e o resultado dessas operações (resto da divisão) é armazenado em duas memórias RAM (uma para cada codificação), cujos tamanhos são de 512 endereços de 32 *bits* cada (16 *kilobits* o que é suficiente para armazenar dados de um grupo de coluna). Quando o processamento do grupo de coluna G termina, inicia-se o processamento do grupo de coluna $G-1$. Os restos das divisões do grupo de coluna G vão sendo lidos das memórias e carregados nos registradores do calculador da divisão, para que seja possível retomar o cálculo da palavra-código. O resto dessas operações novamente é armazenado nas memórias e esse procedimento se repete até o processamento do grupo de coluna 1, cujos resultados das divisões já representam os valores desejados para as paridades intermediárias.

O cálculo das paridades intermediárias nos superquadros do EFEC I.9 modificado é feito da mesma forma que nos superquadros do EFEC I.9 original. Porém, a máquina de estados que controla o processo otimiza o cálculo dos superquadros do EFEC I.9 modificado, diminuindo a latência. Isso é possível pois, conforme mostrado na Figura 5.10, esses superquadros têm 640 colunas à esquerda preenchidas com zeros. Como esses *bits* são processados primeiro, eles não influenciam no cálculo das paridades intermediárias. Então, o cálculo de p_H e p_S no EFEC I.9 modificado é feito processando-se apenas os grupos de coluna 11 ao 2 de seus superquadros *horizontal* e *slope*.

Um dado importante é que os dados de p_H são calculados e encaminhados ao calculador de paridade extra da linha 0 à 511. Já os dados de p_S são entregues da linha 32 à 511 e, em seguida, da linha 0 à 31 devido ao posicionamento dos bits nos superquadros *slope*. Esse fato leva a um desalinhamento no cálculo que tem que ser tratado pelos blocos subsequentes

ao calculador de paridades intermediárias.

5.4.1.4 Calculador de Paridade Extra (Vetores q)

Conforme já foi dito na Seção 5.2.4, os vetores q são calculados para serem um complemento em comum da informação presente em um par de superquadros *horizontal* e *slope*, de tal sorte que os dados calculados para a paridade final (P) sejam válidos para ambos os superquadros.

No pedido de patente intitulado Arquitetura e Método Para Cálculo dos Vetores q em Códigos EFEC [4], que foi depositado no INPI e que é fruto do projeto que originou este trabalho, são mostrados métodos e arquiteturas de circuito para a implementação do cálculo da paridade extra que atendem a diferentes requisitos de latência e estabelecem uma relação de compromisso entre a área (ocupação de elementos lógicos) e latência total. As alternativas de arquitetura e método mostradas no pedido de patente supracitado são baseadas na Equação (5.12), cujos termos $(T.M_H - M_S.T)^{-1}$ e $(p_S - T.p_H)$ podem ser calculados separadamente.

Nas arquiteturas e métodos propostos em [4] e também no presente trabalho, o termo $(p_S - T.p_H)$ é tratado como uma soma das paridades intermediárias chamada de *psum* (soma de p), onde os dados de p_H sofrem entrelaçamento de *bit* antes da operação (representada pelo operador T). A cada par de superquadros *horizontal* e *slope* são calculados 512 vetores de 32 *bits* que compõem a matriz de dados *psum* (volume equivalente a um grupo de coluna), subdividida em 16 blocos de 32 por 32 *bits*.

Pelo fato de haver o desalinhamento entre a produção dos dados de p_H e p_S (comentado anteriormente) nas implementações cobertas por este trabalho, os vetores de p_H correspondentes às linhas 0 a 31 são armazenados em uma memória RAM de 32 endereços de 32 *bits* para serem usados quando chegarem os dados correspondentes de p_S. Os próximos dados de p_H que vão chegando são armazenados em um outro conjunto de 32 memórias de dimensões iguais a 2 endereços de 32 *bits*. Esse conjunto de memórias opera em duas páginas (uma para escrita e outra para leitura) e é usado na execução do entrelaçamento de *bit*. Enquanto isso, os dados de p_S vão sendo armazendados em uma outra memória do tipo FIFO (*first in first out*) para sincronização com os dados de p_H. Uma vez sincronizados, o cálculo de *psum* é executado fazendo-se uma operação ou-exclusivo entre os vetores de 32 *bits* de p_H e p_S da linha 32 até a 512 e em seguida da linha 0 até a 31 (uma operação

por ciclo de relógio). Os dados de *psum* são armazenados em memória para uso na etapa seguinte do cálculo dos vetores q, baseada no termo $(T.M_H - M_S.T)^{-1}$ da Equação (5.12).

Segundo [4], a implementação do termo $(T.M_H - M_S.T)^{-1}$ pode ser feita usando-se matrizes de dados constantes chamadas de *bbb-left*. Tais matrizes independem do conteúdo dos dados de cliente e definem quais *bits* dos vetores resultantes da operação *psum* devem ser usados para cálculo dos vetores q. Para o cálculo de cada *bit* de um vetor q é necessário utilizar um conjunto exclusivo com 1024 *bits* de *bbb-left*, conjunto este que pode ser chamado de bloco ou "matriz de cálculo do *bit* $q_{i,j}$". Também é necessário utilizar um bloco de *psum*, porém esse mesmo bloco será usado para cálculo dos demais *bits* daquele bloco de vetores q. A Figura 5.13 ilustra o cálculo do *bit* 0 do vetor q 0 (vetor q da linha 0 dos superquadros).

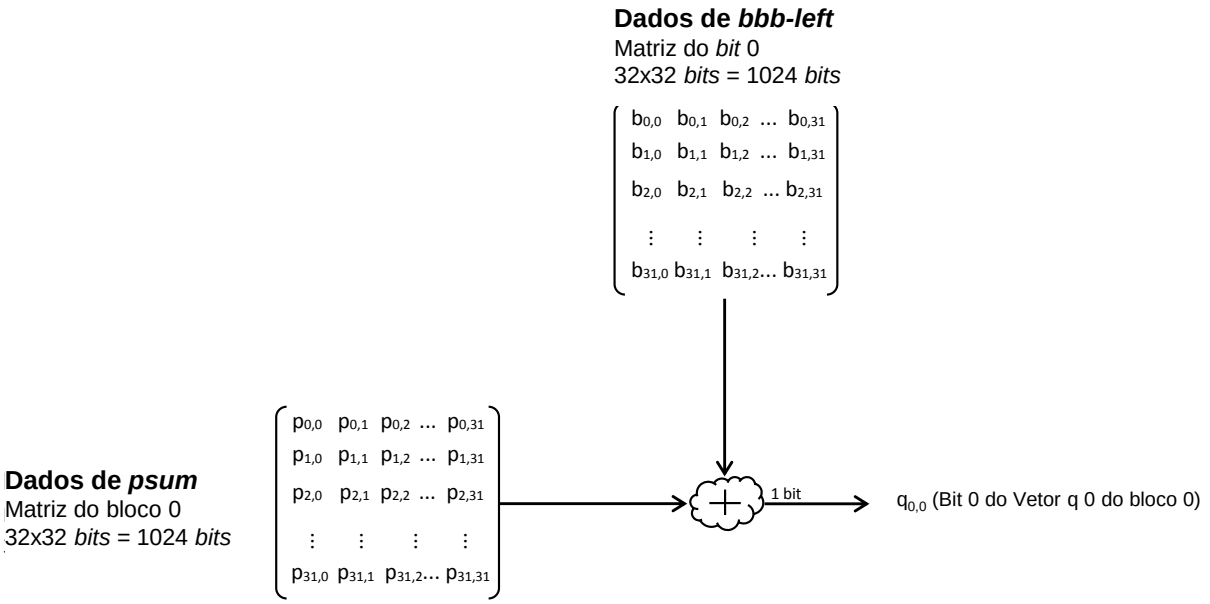

Figura 5.13: Dados envolvidos no cálculo do *bit* 0 do vetor q 0.

Baseado no exemplo retratado pela Figura 5.13, se o elemento na posição $b_{0,0}$ da matriz de *bbb-left* do *bit* 0 for igual a 0, o elemento de mesma coordenada da matriz de *psum* (entenda-se elemento $p_{0,0}$) não influenciará no cálculo do referido *bit*. No entanto, se o elemento na posição $b_{0,0}$ da matriz de *bbb-left* do *bit* 0 for igual 1, então o elemento $p_{0,0}$ entrará no cálculo do vetor q da posição em questão. Depois executada a operação de ou-exclusivo entre todos os *bits*, o resultado será o *bit* $q_{0,0}$ do vetor q 0 do bloco 0 (um *bit* dos 32 que compõem o vetor q da linha 0 de um par *horizontal* e *slope* de superquadros).

Alinhado com a estratégia de modularidade já comentada, foi desenvolvido um circuito denominado Módulo de Cálculo de Vetores q, capaz de efetuar o cálculo simultâneo de 32 *bits* de paridade extra a cada período de 32 ciclos de relógio. Conforme [4], esse circuito

pode ser instanciado mais vezes para ampliar a capacidade de processamento paralelo e, consequentemente, diminuir a latência total causada pelo cálculo dos vetores q. Dentre as alternativas de arquitetura propostas em [4], a escolhida para as implementação do termo $(T.M_H - M_S.T)^{-1}$ está ilustrada na Figura 5.14. Ela utiliza como recursos de *hardware*, além de uma memória RAM que armazena os dados de *psum* (de dimensões iguais a 512 endereços de 32 *bits*), uma memória do tipo ROM (*read-only memory*) de 32 *kilobits* para armazenar 32 matrizes de *bbb-left* iguais às da Figura 5.13 e também dois módulos de cálculo de vetores q em paralelo. Essa arquitetura permite calcular 64 *bits* de paridade extra a cada período de 32 ciclos de relógio. Ou seja, com a arquitetura em questão, essa etapa do cálculo da paridade extra leva cerca de 8192 ciclos de relógio, o que representa em torno de meio superquadro de latência.

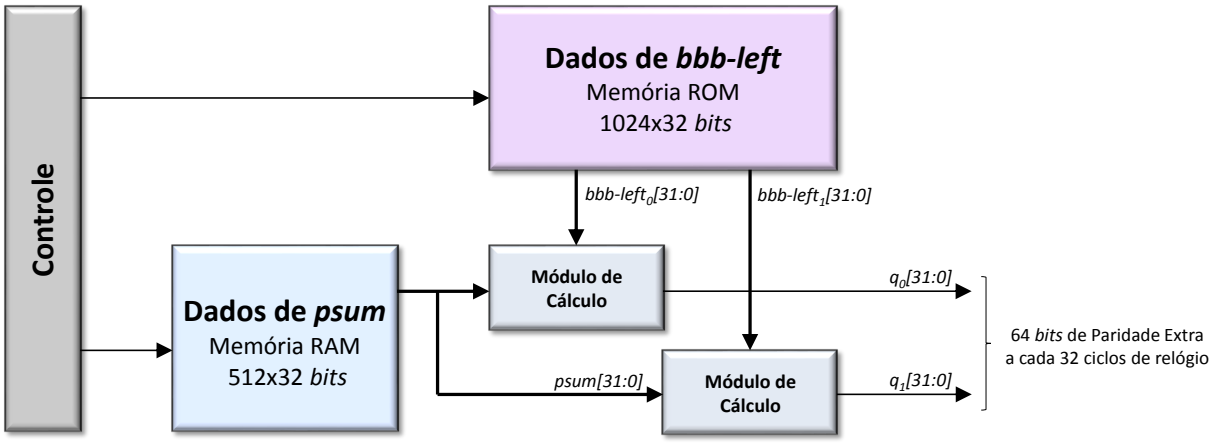

Figura 5.14: Arquitetura da implementação da segunda etapa do cálculo dos vetores q.

5.4.1.5 Calculador de Paridade Final (P)

Depois de calculados os valores de q, o próximo e último passo da codificação é calcular os 512 valores de 32 *bits* da paridade final P comuns aos superquadros *horizontal* e *slope*, mediante o uso da Equação (5.9). O circuito desenvolvido e usado para cálculo das paridades intermediárias (p_H) pôde ser reaproveitado nessa etapa do cálculo, pois a mesma operação é executada em ambos os pontos da arquitetura. É conveniente separar o cálculo em duas etapas. Primeiramente devem ser calculados os resultados referentes ao termo $M_H.q$, que consistem no resto da divisão entre os valores de q pelo polinômio gerador

horizontal $g_H(x)$, apresentado na Equação (5.4), ou seja

$$M_H.q(x) = q(x) \bmod g_H(x). \tag{5.14}$$

Finalmente, os resultados devem ser somados com os valores de paridade intermediária *horizontal* p_H calculados previamente. Trata-se de uma soma módulo-2, implementada com portas lógicas do tipo ou-exclusivo. Os valores calculados de paridade final p são então encaminhados ao bloco memória de paridades e multiplexador de transmissão para que possam ser armazenadas e posteriormente transmitidas.

5.4.1.6 Memória de Paridades e Multiplexador de Transmissão

O bloco memória de paridades e multiplexador de transmissão, como o próprio nome diz, tem duas funções principais: armazenar os valores calculados para as paridades extra e final e gerar os dados já no formato de quadro OTU, agregando o conteúdo do campo FEC. Outra função desse bloco é gerar o pulso de sincronismo da transmissão.

Quando o codificador está desabilitado, os dados ODU (sem FEC) que chegam na entrada do bloco são multiplexados diretamente à saída, pois não há processamento do EFEC. Nesse caso, o multiplexador preenche automaticamente o campo FEC com zeros, conforme reza a Seção 11.1 da recomendação G.709 do ITU-T. Já quando o codificador está habilitado para funcionamento, as paridades extra e final são calculadas e armazenadas normalmente. O multiplexador de transmissão alterna entre esses dados e os dados de cliente (armazenados no bloco memória de dados) para gerar o conteúdo a ser trasmitido, da seguinte maneira: na transmissão do primeiro dos 4 quadros OTU que dão origem aos superquadros EFEC I.9 *horizontal* e *slope*, o multiplexador de transmissão direciona à saída do codificador os dados de cliente correspondentes à linha 1 do quadro OTU (ver Figura 4.3). Ao chegar no campo destinado a FEC, o multiplexador alterna para memória de paridades, que disponibiliza para transmissão os 512 *bits* da coluna mais à esquerda do grupo de coluna que compreende a paridade extra (coluna 63 do superquadro mostrado na Figura 5.3). Os dados da coluna 62 são disponibilizados na sequência, depois os dados das colunas 61 e 60, completando a transmissão da linha 1 do primeiro quadro OTU. O multiplexador volta a selecionar a memória de dados para transmissão da linha 2 e o processo segue assim, alternadamente. Como cada linha de um quadro OTU comporta 256 *bytes* ou 2048 *bits* de dados relacionados a FEC, o que corresponde a 4 colunas do

superquadro EFEC I.9, no primeiro quadro OTU serão transmitidas as colunas 63 a 48 da paridade extra (na segunda linha são alocados os *bits* das colunas 59 a 56 do grupo de coluna 1, na linha 3 são alocados as colunas 55 a 52 e na linha 4 as colunas 51 a 48). No segundo quadro OTU, o restante dos dados da paridade extra são transmitidos (colunas 47 a 32 do grupo de coluna 1). No terceiro e quarto quadros OTU são transmitidos os 512 vetores de 32 *bits* da paridade final (grupo de coluna 0), seguindo a mesma ideia.

Para que os dados das paridades extra e final possam ser lidos em colunas, a memória de paridades é implementada por diversas instâncias de memória em paralelo. Além disso, nesse bloco também é empregada a estratégia de paginação e as capacidades das memórias são diferentes para as implementações do EFEC I.9 original e modificado. No primeiro, a capacidade de armazenamento total é igual a duas páginas de 32 *kilobits* cada (uma página tem o suficiente para armazenar a quantidade de paridade extra e também de paridade final). Já no EFEC I.9 modificado, essa capacidade de armazenamento tem que ser triplicada.

5.4.2 Decodificador

O decodificador é responsável por reorganizar os dados recebidos em formato de quadro OTU nos formatos de superquadro *horizontal* e *slope*, calcular as síndromes e avaliar se a informação foi corrompida durante a transmissão. Em caso positivo, é também função do decodificador processar as síndromes para localizar a posição do(s) erro(s) e efetuar a correção, se estiver dentro de suas capacidades.

A Figura 5.15 é um diagrama de blocos que mostra a arquitetura de alto nível usada para implementação do decodificador EFEC I.9 original. Conforme denota a Figura 5.16, a principal diferença entre essa arquitetura e a que foi desenvolvida para implementação do EFEC I.9 modificado é que a segunda tem três instâncias em paralelo do bloco processador ao invés de apenas uma. Dado que as etapas de processamento de síndromes, localização e correção de erros são custosas em termos de tempo de execução, inclusive pelo fato de serem realizadas iterativamente por até trinta vezes, torna-se necessário lançar mão do processamento simultâneo das mesmas nos superquadros 1, 2 e 3 das codificações *horizontal* e *slope*. A latência total ocasionada pela decodificação é da ordem de dois superquadros.

Os dados ODU chegam ao decodificador em um barramento paralelo de 32 *bits* de largura e chegam a três pontos: o sub-bloco memória de dados localizado dentro do pro-

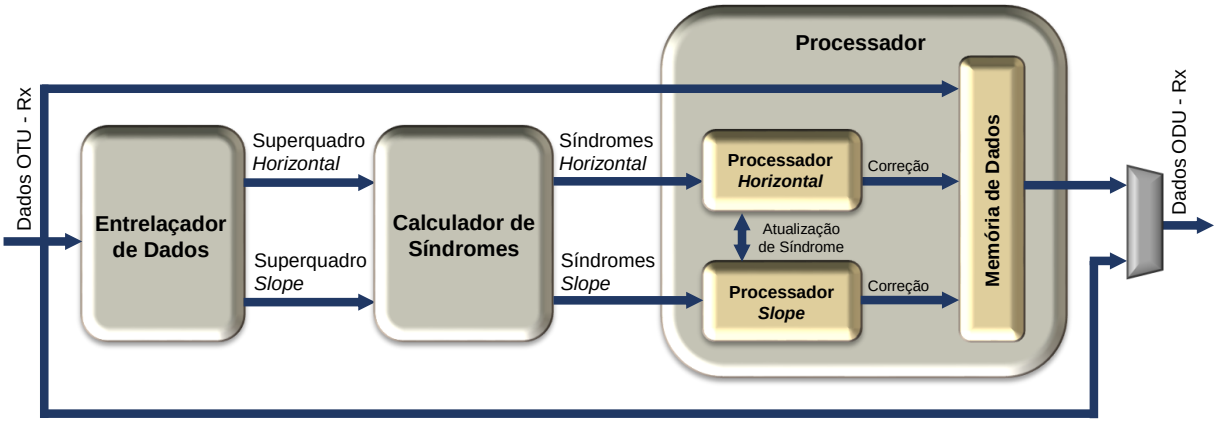

Figura 5.15: Diagrama de blocos do decodificador EFEC I.9 original.

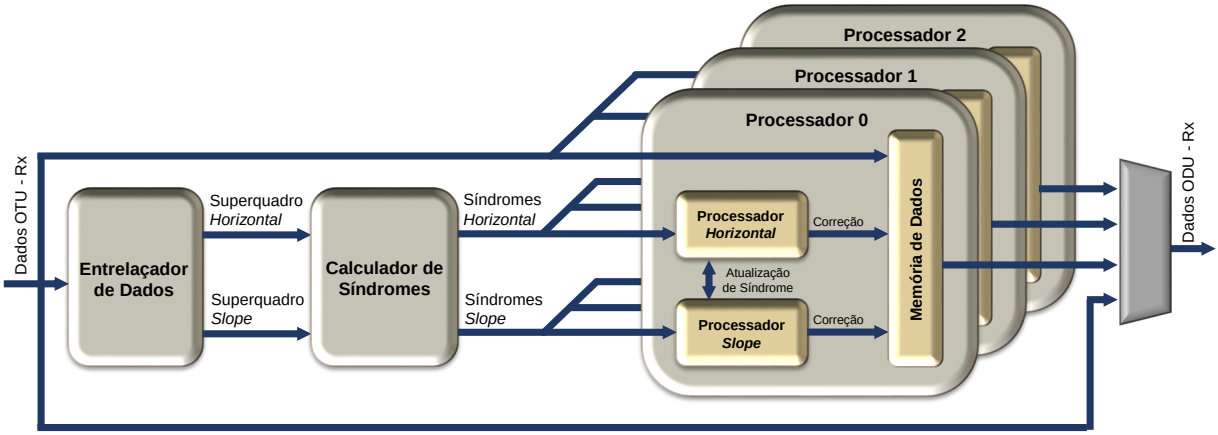

Figura 5.16: Diagrama de blocos do decodificador EFEC I.9 modificado.

cessador (ou os sub-blocos memória de dados no caso da implementação do EFEC I.9 modificado), o bloco entrelaçador e o multiplexador de saída. Quando o decodificador está desabilitado, o multiplexador direciona os dados OTU da entrada diretamente para a saída.

5.4.2.1 Entrelaçador

A exemplo do entrelaçador do codificador, o bloco homônimo do decodificador tem por função entregar os dados nos formatos dos superquadros *horizontal* e *slope* para a próxima etapa do fluxo de processamento. Todavia, no decodificador os dados chegam em formato de quadro OTU ao invés de ODU, pois estão inclusos os dados de FEC que foram calculados no codificador.

A implementação do bloco em questão foi realizada com recursos lógicos muito semelhantes aos usados no entrelaçador do codificador, ou seja, máquinas de estados, memórias

RAM e multiplexadores. A diferença reside justamente na necessidade de armazenamento dos dados referentes a FEC. No codificador, O campo de FEC chega preenchido com zeros, que obviamente não precisam ser armazenados. No caso do decodificador, chegam 256 *bytes* do campo de FEC a cada linha de um quadro OTU e os mesmos só serão usados no fim do cálculo de síndrome (porque constituem os grupos de coluna 0 e 1). À vista disso, o armazenamento se faz necessário.

O cálculo das síndromes é realizado com uma dinâmica cíclica semelhante à do cálculo das paridades intermediárias p_H e p_S explicada na Subseção 5.4.1.2, de sorte que a Figura 5.12 também pode ser usada aqui como ilustração. Tal e qual no codificador, essa estratégia de cálculo possibilita economizar principalmente o recurso lógico memória RAM, já não é necessário armazenar todo o superquadro para efetuar o entrelaçamento. Em cada ciclo, um grupo de coluna é armazenado em memórias RAM paginadas, enquanto o grupo de coluna que foi armazenado no ciclo anterior é lido e disponibilizado em palavras de 32 *bits* por palavra-código ao calculador de síndromes para processamento. Mais uma vez, os conhecimentos citados na patente [6] foram aproveitados.

5.4.2.2 Calculador de Síndromes

Segundo [26], a distância de Hamming d entre duas palavras-código é número de posições nas quais seus elementos diferem. A distância mínima d_{min} de um código é a menor distância de Hamming entre duas de suas palavras-código. Um código cuja distância mínima é igual a d_{min} é capaz de corrigir até t erros se e somente se

$$t \leq \lfloor 1/2(d_{min} - 1) \rfloor. \tag{5.15}$$

Para encontrar e corrigir os erros introduzidos em uma palavra-código BCH, define-se as síndromes $S_1, S_2, S_3, ..., S_{2t}$. As síndromes são definidas como a avaliação do polinômio recebido $r(x)$ em $x = \alpha, \alpha^2, \alpha^3, ..., \alpha^{2t}$, tal que

$$S_i = r(\alpha^i) = e(\alpha^i), \tag{5.16}$$

para $i = 1, 2, 3, ..., 2t$.

Percebe-se pela Equação (5.16) que as síndromes dependem apenas do polinômio de erro e não da palavra-código em si. Portanto, se nenhum erro for introduzido durante a

transmissão, todas as síndromes serão iguais a zero. Se o número de erros introduzidos em uma palavra-código for igual ou menor que t, as síndromes conterão informação suficiente para corrigir os erros. Para os códigos EFEC deste trabalho, tem-se que t é igual a 3, o que significa que tais códigos são capazes de corrigir até 3 erros por palavra-código e seis síndromes devem ser calculadas. Porém, uma vez que $S_2 = S_1^2$, $S_4 = S_2^2$ e $S_6 = S_3^2$, pode-se calcular apenas as síndromes independentes S_1, S_3 e S_5. Considerando a introdução de j erros a uma palavra-código e as j posições de erro como sendo e_1, e_2, e_3, ..., e_j, o polinômio de erro pode ser representado por

$$e(x) = \sum_{i=1}^{j} x^{e_t}, \tag{5.17}$$

onde $0 \leq e_i \leq n-1$. Para o caso onde $j = 3$, as síndromes são dadas por

$$S_1 = \quad \alpha^{e_1} + \alpha^{e_2} + \alpha^{e_3}; \tag{5.18a}$$

$$S_3 = \quad \alpha^{3e_1} + \alpha^{3e_2} + \alpha^{3e_3}; \tag{5.18b}$$

$$S_5 = \quad \alpha^{5e_1} + \alpha^{5e_2} + \alpha^{5e_3}. \tag{5.18c}$$

As síndromes S_1, S_3 e S_5 correspondentes às palavras-código das codificações *horizontal* e *slope* são valores de 10 *bits* cada, calculados para todos os superquadros e disponibilizados ao bloco processador, no caso do EFEC I.9 original, ou aos blocos processadores, no caso do EFEC I.9 modificado. Dito cálculo é desempenhado por multiplicadores que usam o esquema de Horner [27] no campo de Galois e armazenados em memória RAM para serem acessados novamente durante o próximo ciclo de cálculo, no qual são processados os dados do grupo de coluna imediatamente à direita. Um vetor de dados recebido (palavra-código de tamanho n adicionada de possíveis erros) pode ser escrito sob a forma

$$a(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0. \tag{5.19}$$

As síndromes S_1, S_3 e S_5 da codificação *horizontal* são calculadas avaliando-se suas palavras-código nos elementos primitivos α, α^3 e α^5 do campo de Galois, respectivamente. Para se calcular as síndromes *slope* usando o mesmo multiplicador empregado no caso das síndromes *horizontal*, o cálculo tem que ligeiramente adaptado. Os vetores de dado são recebidos como em (5.19). Porém, deseja-se avaliar os polinômios $a'(x) = x^{n-1}a(x^{-1})$.

Então a Equação (5.19) é reescrita da seguinte maneira

$$\begin{aligned} x^{n-1}a(x^{-1}) = & \quad x^{n-1}(a_{n-1}x^{-(n-1)} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0) \\ = & \quad x^{n-1}(a_0 + x^{-1}(a_1 + \cdots + x^{-1}(a_{n-2} + a_{n-1}x^{-1})\cdots)), \end{aligned} \tag{5.20}$$

o que significa que as síndromes S_1, S_3 e S_5 para a codificação *slope* podem ser calculadas pelo esquema de Horner, mas os polinômios que representam suas palavras-código recebidas adicionadas de erro têm que ser avaliados nos elementos α^{-1}, α^{-3} e α^{-5} do campo de Galois. Além disso, é necessária ainda uma multiplicação adicional pelo termo x^{n-1} como se pode ver na Equação (5.20).

Além de efetuar o cálculo dos valores de síndrome, o bloco calculador de síndromes tem outra função relevante. Recorrendo aos termos (x^2+1) e (x^2+x+1) dos polinômios geradores $g_H(x)$ e $g_S(x)$, respectivamente, ambos citados na Subseção 5.2.1, o bloco em questão calcula também dois *bits* de paridade adicional por palavra-código. Os referidos *bits* são usados pelo processador para reduzir a probabilidade média de falsa detecção de erro.

5.4.2.3 Processador

As funções essenciais do processador são: armazenar os valores de síndrome, analisá-los a fim de descobrir se as palavras-código foram recebidas com erro(s) e, em caso positivo, determinar as localizações do(s) mesmo(s), representadas respectivamente por α^{e_1}, α^{e_2} e α^{e_3}, e atuar na correção das palavras-código. Os recursos aplicados para tanto incluem, entre outros, máquinas de estados e memórias dos tipos RAM e FIFO.

Quando os erros introduzidos durante a transmissão são representados pelo polinômio resultante da Equação (5.17), pode ser formado o polinômio localizador de erros

$$\sigma(x) = (x + \alpha^{e_1})(x + \alpha^{e_2})(x + \alpha^{e_3}). \tag{5.21}$$

Expandindo-se a Equação (5.21) e denotando-se os coeficientes de $\sigma(x)$ como σ_1, σ_2 e σ_3, obtém-se

$$\sigma(x) = x^3 + \sigma_1 x^2 + \sigma_2 x + \sigma_3. \tag{5.22}$$

De acordo com [28], a Equação (5.22) pode ser solucionada de forma analítica e a

localização dos erros obtida diretamente dos coeficientes do polinômio localizador de erros $\sigma(x)$, o que elimina a necessidade de se usar um algoritmo de busca de Chien [29] e reduz o número de operações matemáticas quando comparado com esse tipo de algoritmo. Desse modo, a decodificação executada pelo bloco processador é fundamentada em reduções do polinômio localizador de erro $\sigma(x)$, recorrendo-se a transformações [30] para reduzir seu grau de cúbico para quadrático e posteriormente solucionando as equações resultantes usando as técnicas mostradas em [31].

Usando as identidades de Newton [21], pode-se mostrar que os coeficientes σ_1, σ_2 e σ_3 são relacionados às síndromes, tal que

$$\sigma_1 = S_1; \tag{5.23a}$$

$$\sigma_2 = \frac{S_5+S_3S_1^2}{S_3+S_1^3}; \tag{5.23b}$$

$$\sigma_3 = S_1^3 + \frac{S_3^2+S_1S_5}{S_3+S_1^3}. \tag{5.23c}$$

Desta forma, para encontrar a localização dos erros, resolve-se a equação cúbica

$$\sigma(x) = 0 \Longleftrightarrow x^3 + \sigma_1 x^2 + \sigma_2 x + \sigma_3 = 0 \tag{5.24}$$

submetendo a mesma a uma transformação específica que depende dos valores das síndromes. Três transformações diferentes são aplicadas na abordagem deste trabalho em particular e cada uma resulta em polinômios no campo de Galois $GF(2^{10})$, nos quais apenas os coeficientes dos termos de grau zero variam. Todas as raízes destes novos polinômios foram calculadas previamente e implementadas sob a forma de tabelas cujo índice é o coeficiente do termo de grau zero, de jeito que possam ser acessadas durante o processo de decodificação.

Algumas combinações de valores de síndromes indicam o número de erros possivelmente introduzidos na palavra-código e abordagens diferentes são tomadas baseadas nesses

valores. Quatro expressões condicionais são empregadas para determinar o plano de ação:

$$S_1 = S_3 = S_5 = 0; \tag{5.25a}$$

$$S_1^3 = S_3; \tag{5.25b}$$

$$S_1^5 = S_5; \tag{5.25c}$$

$$\sigma_3 = 0. \tag{5.25d}$$

A avaliação de tais expressões leva aos cenários descritos a seguir, onde a localização dos erros são denotadas por x_1, x_2 e x_3.

- 0 erro

  Quando nenhum erro acometer a palavra-código, todas as suas síndromes serão iguais a 0, isto é, $S_1 = S_3 = S_5 = 0$. Nesse caso, nenhuma ação é requerida.

- 1 erro

  Tal como já foi falado, as síndromes dependem apenas do polinômio de erro e não da palavra código em si. Quando a palavra-código contiver apenas um erro, pode-se provar que este erro leva a

  $$S_1^3 = S_3 \wedge S_1^5 = S_5 \tag{5.26}$$

  e a localização do erro é dada por

  $$x_1 = \sigma_1 = S_1. \tag{5.27}$$

- 2 erros

  Quando dois erros são introduzidos na palavra-código pode-se mostrar que $\sigma_3 = 0$. A Equação (5.24) fica, portanto, reduzida a $x^2 + \sigma_1 x + \sigma_2 = 0$ e, usando-se a transformação $x = y\sigma_1$, obtém-se

  $$y^2 = y + k_3, \quad \textit{onde} \quad k_3 = \frac{\sigma_2}{\sigma_1^2}. \tag{5.28}$$

  Expressando-se as duas raízes da Equação (5.28) como y_{31} e y_{32}, as localizações dos

erros são dadas por

$$x_1 = y_{31}\sigma_1; \tag{5.29a}$$

$$x_2 = y_{32}\sigma_1. \tag{5.29b}$$

- 3 erros (cenário 1)

  Se $\sigma_1 = \sqrt{\sigma_2} \Longleftrightarrow S_1^5 = S_5$ a transformação $x = y + \sigma_1$ pode ser usada para reduzir a Equação (5.24) a

$$y^3 + k_2 = 0, \quad onde \quad k_2 = \sigma_1^3 + \sigma_3. \tag{5.30}$$

  Denotando-se as três raízes da Equação (5.30) como y_{21}, y_{22} e y_{23}, as localizações dos erros são dadas por

$$x_1 = y_{21} + \sigma_1; \tag{5.31a}$$

$$x_2 = y_{22} + \sigma_1; \tag{5.31b}$$

$$x_3 = y_{23} + \sigma_1. \tag{5.31c}$$

- 3 erros (cenário 2)

  Se $\sigma_1 \neq \sqrt{\sigma_2} \Longleftrightarrow S_1^5 \neq S_5$ a transformação $x = \sigma_1 + y(\sigma_1 + \sqrt{\sigma_2})$ pode ser usada para reduzir a Equação (5.24) a

$$y^3 + y + k_1 = 0, \quad onde \quad k_1 = \frac{\sigma_1\sigma_2 + \sigma_3}{(\sigma_1 + \sqrt{\sigma_2})^3}. \tag{5.32}$$

  Denotando-se as três raízes da Equação (5.32) como y_{11}, y_{12} e y_{13}, as localizações dos erros são

$$x_1 = \sigma_1 + y_{11}(\sigma_1 + \sqrt{\sigma_2}); \tag{5.33a}$$

$$x_2 = \sigma_1 + y_{12}(\sigma_1 + \sqrt{\sigma_2}); \tag{5.33b}$$

$$x_3 = \sigma_1 + y_{13}(\sigma_1 + \sqrt{\sigma_2}). \tag{5.33c}$$

- 4 erros ou mais

Quando uma palavra-código recebida possui 4 erros ou mais, a quantidade de erros supera a capacidade de correção do código. Sempre que

$$S_1^3 = S_3 \wedge S_1^5 \neq S_5, \tag{5.34}$$

onde $\wedge$ é um conectivo lógico que representa a operação "e", pode-se concluir que a palavra-código recebida contém 4 erros ou mais, porque a Equação (5.34) não será verdadeira nem para os casos de 0 e 1 erro (já que $S_1^5 = S_5$ seria verdade em ambos os casos), nem tampouco para os casos de 2 e 3 erros, pelas razões relatadas a seguir.

No caso de 2 erros, se $e(x) = x^{e_1} + x^{e_2}$, tem-se $S_1 = \alpha^{e_1} + \alpha^{e_2}$ e $S_3 = \alpha^{3e_1} + \alpha^{3e_2}$. Então $S_1^3 + S_3 = (\alpha^{e_1} + \alpha^{e_2})^3 + \alpha^{3e_1} + \alpha^{3e_2} = \alpha^{e_1}\alpha^{e_2}(\alpha^{e_1} + \alpha^{e_2}) \neq 0$ e, portanto, $S_3 \neq S_1^3$.

No caso de 3 erros, se $e(x) = x^{e_1} + x^{e_2} + x^{e_3}$, tem-se $S_1 = \alpha^{e_1} + \alpha^{e_2} + \alpha^{e_3}$ e $S_3 = \alpha^{3e_1} + \alpha^{3e_2} + \alpha^{3e_3}$. Então:

$$\begin{aligned} S_1^3 + S_3 &= (\alpha^{e_1} + \alpha^{e_2} + \alpha^{e_3})^3 + \alpha^{3e_1} + \alpha^{3e_2} + \alpha^{3e_3} \\ &= \alpha^{e_1}\alpha^{2e_2} + \alpha^{e_1}\alpha^{2e_3} + \alpha^{e_2}\alpha^{2e_3} + \alpha^{e_2}\alpha^{2e_1} + \alpha^{e_3}\alpha^{2e_1} + \alpha^{e_3}\alpha^{2e_2} \\ &= \begin{vmatrix} 1 & \alpha^{e_1} & \alpha^{2e_1} \\ 1 & \alpha^{e_2} & \alpha^{2e_2} \\ 1 & \alpha^{e_3} & \alpha^{2e_3} \end{vmatrix} \neq 0 \end{aligned}$$

Portanto, $S_3 \neq S_1^3$.

Outros padrões com mais de 3 erros produzem síndromes com valores que se encaixam nas características dos padrões com 3 ou menos erros. Alguns desses padrões fazem com que o processador corrija a palavra-código indevidamente, adicionando mais erros à mesma. Entretanto, há padrões que resultam em valores de k que, por sua vez, levam a equações polinomiais em conformidade com (5.28), (5.30) e (5.32) cuja quantidade de raízes em $GF(2^{10})$ não é a esperada. Este fato propicia ao processador detectar que há mais de 3 erros na palavra-código. Outro recurso que auxilia nesse sentido são os dois *bits* de paridade adicional mencionados na Subseção 5.4.2.2, responsáveis por diminuir a probabilidade média de detecção falsa de erro de 1/6 para 1/24.

À semelhança de outros códigos concatenados, o EFEC I.9 dispõe da decodificação iterativa; uma característica que pode oportunizar a correção completa de uma palavra-código que foi recebida com mais de 3 erros. Para ilustrar tal recurso juntamente com o processo de decodificação, propõe-se os superquadros *horizontal* e *slope* apresentados na Figura 5.17, ambos montados pelo bloco entrelaçador a partir de um mesmo conjunto de dados recebidos que também é armazenado no sub-bloco memória de dados, posicionado dentro do processador.

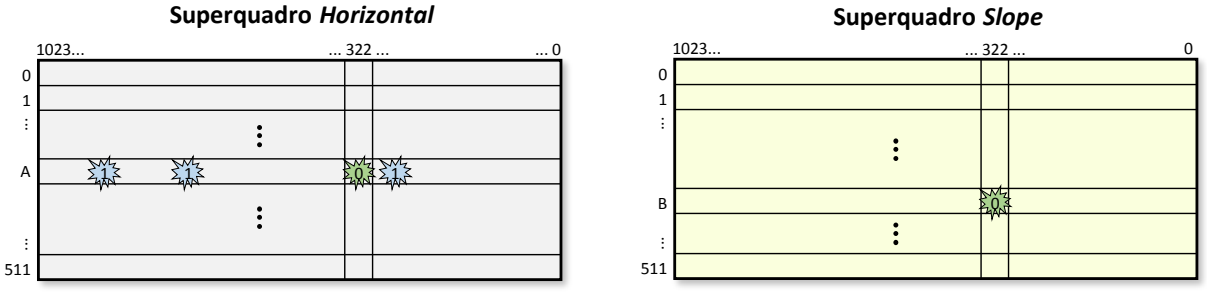

Figura 5.17: Exemplo de superquadros montados a partir de dados recebidos com erro.

Como exemplo, considere a decodificação das linhas A do superquadro *horizontal* e B do superquadro *slope*. Na linha A encontram-se 4 *bits* cujos valores foram invertidos durante a transmissão, ou seja, 4 erros. Os mesmos 4 *bits* foram distribuídos em outras linhas do superquadro *slope*, porém suas posições de coluna permanecem inalteradas consoante às regras de entrelaçamento explicadas na Subseção 5.2.3 deste documento e também na Subseção I.9.2.3 de [3]. Suponha, então, que o *bit* que se encontra na coordenada hipotética linha A, coluna 322 do superquadro *horizontal* foi alocado na mesma coluna da linha B no superquadro *slope* onde só há esse *bit* errado.

As palavras-código vão sendo disponibilizadas ao bloco calculador de síndromes, que calcula os valores de S_1, S_3, S_5 e dos dois *bits* de paridade adicional para cada linha (palavra-código) dos superquadros *horizontal* e *slope* e disponibiliza os resultados para que sejam armazenados nas memórias de síndrome (memórias RAM instanciadas dentro dos processadores de cada codificação).

Conforme ilustrado na Figura 5.18, o processamento começa sempre pelo processador *horizontal* (primeira iteração *horizontal*), que lê o valor das síndromes da linha 0 até a 511. Ao avaliar os valores correspondentes à linha A da forma como foi explanado anteriormente nesta subseção, o processador percebe que há mais erros do que ele é capaz de corrigir em uma única iteração e o processamento dessa linha é interrompido sem que seja tomada

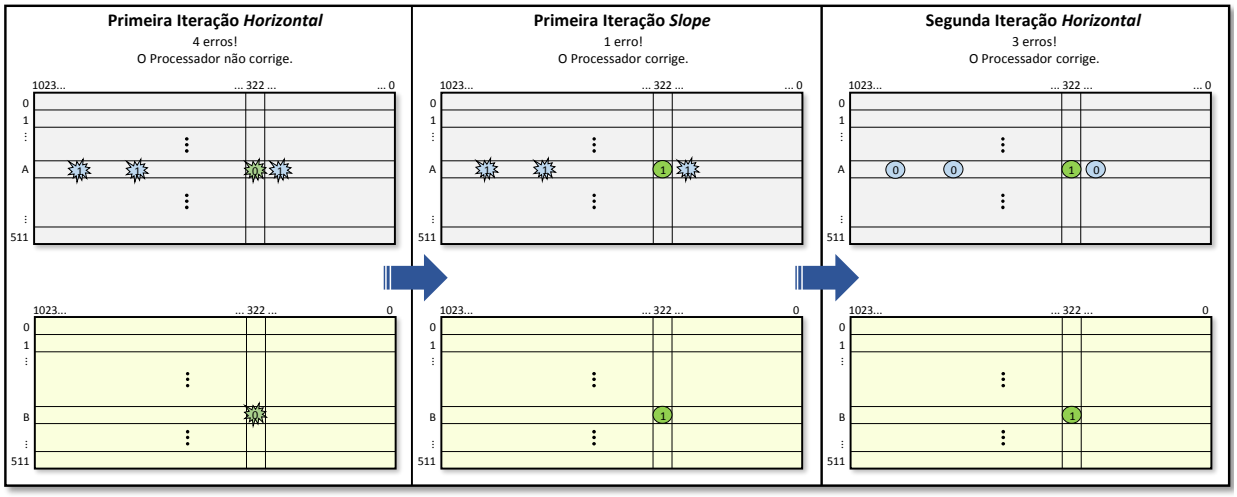

Figura 5.18: Caso de correção com decodificação iterativa.

qualquer ação.

Depois de avaliar todas as síndromes da codificação *horizontal*, inicia-se a primeira iteração *slope*. Na vez da linha B, o processador avalia que há um único erro e dispara os cálculos para localização do mesmo. Concluídos os cálculos, o processador sabe que a posição da memória de dados que abriga o *bit* correspondente à coordenada linha B, coluna 322 do superquadro *slope* contém um erro. No caso do EFEC I.9 original a correção é executada na hora. Como o EFEC I.9 modificado tem três blocos processadores em paralelo para cada codificação, aquele que encontrou o erro verifica se algum outro processador está acessando a memória. Em caso positivo, coloca a correção em uma memória FIFO para que ela seja feita assim que for possível. Em paralelo com a inversão do *bit* na memória de dados, o processador atualiza o valor da síndrome da linha B na memória de síndromes *slope*, que fica com conteúdo correspondente a essa linha zerado. Atualiza também o valor da síndrome da linha A na memória de síndromes *horizontal*, que passa a ter conteúdo que representa três erros ao invés de quatro.

Ao ser concluída a primeira iteração *slope*, inicia-se a segunda iteração *horizontal*. Dessa vez, a avaliação das síndromes da linha A resulta que existem em 3 erros na mesma. Dado que essa quantidade está dentro de sua capacidade de correção, o processador dispara os cálculos de localização dos erros. Após determinadas as localizações em questão, a memória de dados e as memórias de síndrome *horizontal* e *slope* são atualizadas.

O processo de decodificação iterativa segue até que todas as síndromes de ambas as codificações sejam zeradas ou até que se inicie o processamento de um novo conjunto de

dados. Pelo que foi dito, conclui-se que essa técnica de decodificação contribui para o alto desempenho dos EFECs deste trabalho, pois viabiliza a correção parcial ou completa dos dados de uma palavra-código a despeito do fato de que esta apresenta quantidade de erros superior à capacidade de correção do BCH(1020,988).

Capítulo 6

Arranjo Experimental e Resultados

Para que fosse possível testar e colher resultados das implementações dos EFECs citadas nos capítulos anteriores, um ambiente de testes tal qual o ilustrado no diagrama de blocos da Figura 6.1 foi montado em laboratório.

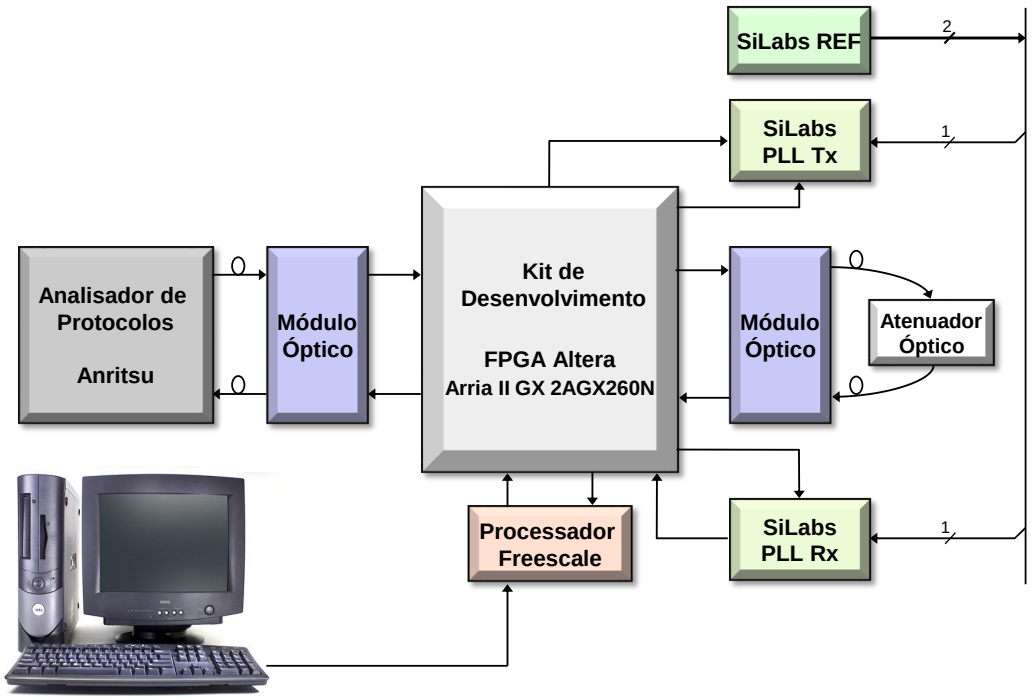

Figura 6.1: Ambiente de testes.

Os códigos foram compilados, sintetizados e gravados em um componente de lógica programável FPGA do fabricante Altera, modelo Arria II GX 2AGX260N, que equipa o *kit* de desenvolvimento da mesma marca, modelo DK-DEV-2AGX260N. Para tais atividades, o *software* Altera Quartus II 64-*Bit*, versão 13.0.1 SP1 (*service pack* 1) foi utilizado.

O ambiente de testes conta também com um microcomputador conectado a um *kit* da Axiom Manufacturing, modelo CMM-52259 AXM-0515, equipado com um processador Freescale Coldfire 52259, no qual foi gravado um *software* de interface de linha de comando (CLI, *command-line interface*) desenvolvido em linguagem C++. Por meio desses equipamentos é possível monitorar e configurar os EFECs, acessando-se registros internos implementados nos blocos de controle do codificador e do decodificador.

Outro equipamento que faz parte do ambiente de testes é um sintetizador de relógio Silicon Labs, modelo Si5338, que equipa o *kit* do mesmo fabricante, modelo Si5338-EVB. Tal sintetizador gera relógios de referência para dois PLLs (*phase locked loop*) também da Silicon Labs, modelo Si5326, que equipam dois *kits* Si5325/26-EVB. A função dos PLLs é fornecer os relógios para a operação do codificador e do decodificador, com a mesma frequência porém independentes, contribuindo assim para uma maior similiaridade entre o ambiente de testes e o ambiente real.

O sinal cliente é gerado em meio óptico pelo analisador de protocolos da marca Anritsu, modelo MP1570A, sob a forma de sequência pseudoaleatória ou PRBS (*pseudorandom binary sequence*). Para o teste do EFEC I.9 original, os dados são gerados a uma taxa de 2,66 Gbps, enquanto que para os testes do EFEC I.9 modificado, a taxa é de 3,01 Gbps. A taxa de sinal cliente, portanto, é de 2,488 Gbps para ambos os casos. Um módulo óptico converte o sinal gerado pelo equipamento Anritsu para o meio elétrico antes de ser aplicado no *kit* de desenvolvimento Altera, onde passa por todo o processamento do lado da transmissão. Em seguida, o sinal é convertido novamente para o meio óptico, passa por um atenuador óptico do fabricante Hewlett-Packard, modelo 8156A, e é convertido novamente para o meio elétrico para voltar a ser inserido no *kit* de desenvolvimento Altera, onde agora sofre o processamento do lado da recepção. Após tal processamento, o sinal deixa novamente o *kit* Altera, é convertido para o meio óptico e retorna ao analisador Anritsu para medições de taxa de erro. A Figura 6.2 mostra uma visão geral do ambiente montado em laboratório. Já a Figura 6.3 mostra em detalhes a montagem dos *kits* com o FPGA, o gerador de relógio de referência, dos PLLs, do processador e também os módulos ópticos.

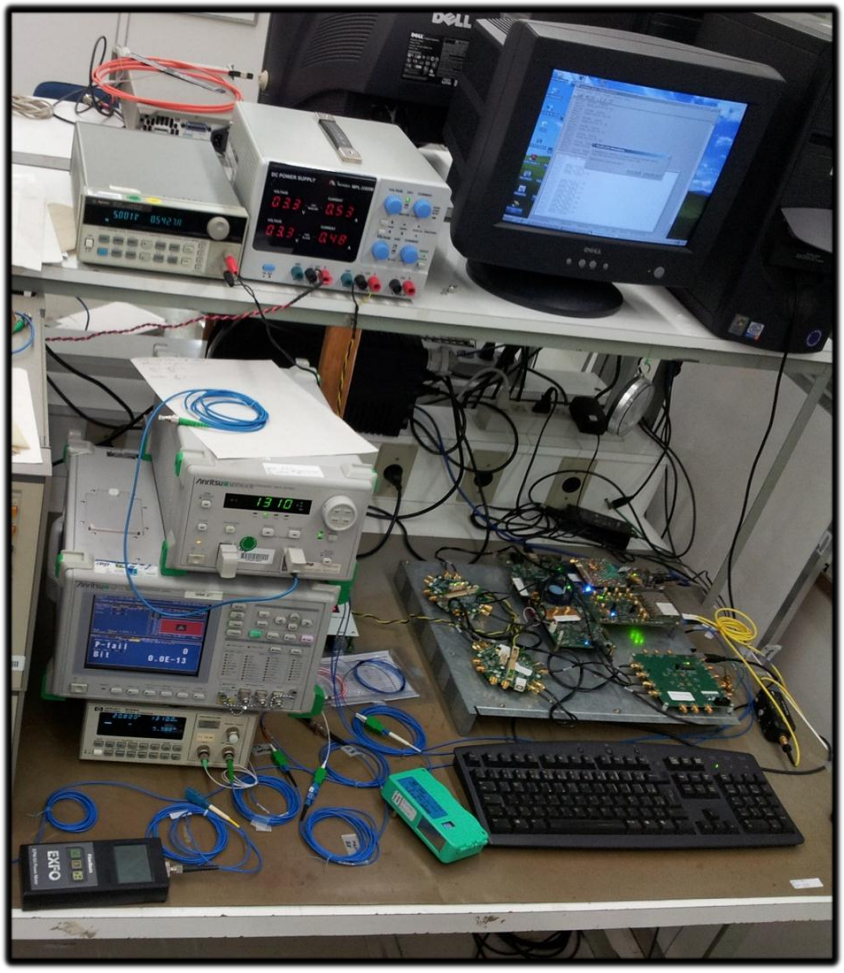

Figura 6.2: Visão geral do ambiente de testes.

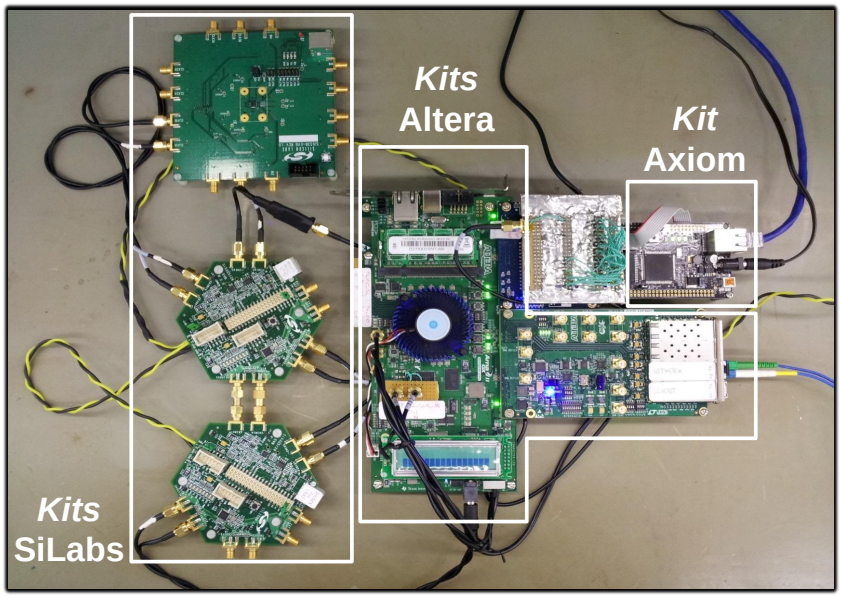

Figura 6.3: *Kits* utilizados no ambiente de testes.

6.1 Arquitetura Interna de Projeto no FPGA

A Figura 6.4 é um diagrama de blocos que retrata a arquitetura interna do projeto no FPGA. Como se pode notar, outros blocos além do codificador EFEC e do decodificador EFEC foram implementados para tratar o fluxo de dados e dar suporte aos testes.

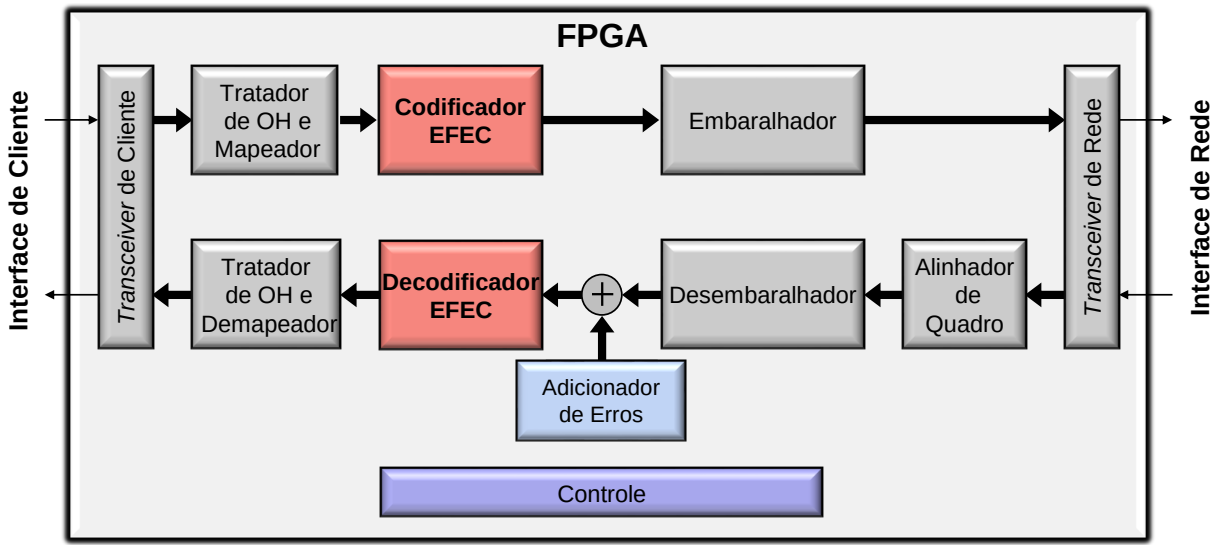

Figura 6.4: Diagrama de blocos da arquitetura interna de projeto no FPGA.

O projeto faz uso de uma estrutura simplificada do quadro OTU, na qual são excluídos os tratamentos do cabeçalho ODU e de justificação de *bits*. O sinal do cliente é mapeado diretamente no quadro OPU, de forma síncrona (CBR, *constant bit rate*), portanto não é feita a terminação do quadro de cliente. As seguintes funcionalidades dos cabeçalhos OTU e OPU foram implementadas conforme [2]:

- **Alinhamento de quadro e multiquadro** (campos FAS e MFAS do cabeçalho OTU):

  Localizados nos sete primeiros *bytes* de cada quadro, os dados de FAS e MFAS são preenchidos pelo bloco tratador de OH e mapeador. O campo FAS é a única parte do quadro OTU que não sofre processo de embaralhamento pelo bloco embaralhador. A palavra de alinhamento de quadro (FAS) é composta pelos seis *bytes* $0xF6F6F6282828$, enquanto a palavra de alinhamento de multiquadro (MFAS) é composta por um único *byte*. Os valores de MFAS são gerados por meio de um contador de 8 *bits* que disponibiliza em sua saída valores crescentes e sequenciais de $0x00$ a $0xFF$ (incrementados de uma unidade a cada quadro) possibilitando, assim, a identificação dos quadros de um determinado multiquadro.

No fluxo de dados de recepção, o bloco alinhador de quadro busca constantemente pela sequência de alinhamento de quadro (FAS) para que seja possível realizar a sincronização de *byte* dos dados recebidos e acusar quando há perda de sincronismo de quadro. O bloco tratador de OH e demapeador monitora a palavra de alinhamento de multiquadro (MFAS) por intermédio de uma máquina de estados cuja implementação segue [32]. O mesmo bloco também acusa quando há perda de sincronismo de multiquadro.

- ***Section monitoring*** (campo SM do cabeçalho OTU):
  Foram implementados os seguintes subcampos: BIP-8 *(bit interleaved parity-8)*, BDI *(backward defect indication)*, BEI *(backward error indication)* e TTI *(trail trace identifier)*.

- ***General communications channel*** (campo GCC0 do cabeçalho OTU):

  O canal GCC0 permite o envio de informações de gerência e sinalização entre elementos da rede OTN. Segundo [32], seu uso é opcional.

  É possível controlar esta funcionalidade por meio de registradores localizados no bloco de controle. Foi implementado um registrador que habilita ou desabilita o uso do canal GCC0, fazendo com que o campo seja transmitido com conteúdo zerado quando a funcionalidade está desabilitada. Um outro registrador configura o modo de operação. Em modo normal, o canal GCC0 ocupa o campo GCC0 (*bytes* 11 e 12) do cabeçalho OTU, enquanto no modo estendido são ocupados os campos GCC0 (*bytes* 11 e 12) e RES (*bytes* 13 e 14) do cabeçalho OTU, provendo o dobro da taxa deste canal em relação ao modo normal.

- ***Payload structure identifier*** (campo PSI do cabeçalho OPU):

  No subcampo PT (*payload type*) o tipo do sinal de cliente é indicado como sendo do tipo CBR pelo valor em hexadecimal $0x0021$. O subcampo CSF (*client signal fail*) também foi inplementado.

Além das funcionalidades listadas anteriormente, as indicações dos seguintes defeitos (alarmes da rede OTN) são geradas e disponibilizadas em registros para leitura: dLOF (*loss of frame*), dLOM (*loss of multiframe*), dTIM (*trace identifier mismatch*), dBDI (*backward*

defect indication), sendo que a ocorrência de qualquer um dos três primeiros também gera pedido de interrupção ao processador por intermédio do bloco de controle.

Analisando-se a Figura 6.4, é possível notar que os dados são inseridos serialmente no FPGA e transformados em um barramento paralelo de 16 *bits* de largura pelo *transceiver* de cliente. Em seguida, um sub-módulo executa o tratamento do cabeçalho (OH, *overhead*) do quadro OTU1 e também o mapeamento em quadro ODU1. Nesse processo, o sinal de cliente não é desencapsulado, sendo somente amostrado em taxa de *bit* constante (CBR, *constant bitrate*) e mapeado diretamente no quadro, segundo as recomendações do ITU-T G.709 [2] e G.798 [32].

O próximo bloco no fluxo de transmissão é o codificador EFEC, que processa os dados de cliente, calcula as paridades e gera o quadro OTU1, preenchendo a área destinada a FEC. O bloco embaralhador (*scrambler*) completa o fluxo de transmissão, embaralhando os dados segundo a Seção 11.2 do documento G.709 [2]. Após passar pelo *transceiver* de rede, os dados estão em formato serial, prontos para transmissão.

No fluxo de recepção, primeiramente os dados são paralelizados pelo *transceiver* de rede, iniciando o fluxo de recepção. Em seguida, é a vez do alinhador de quadro tratar os dados, encontrando a sequência de alinhamento e alinhando o quadro segundo descrito na Seção 8.2 da recomendação G.798 [32]. Após esse procedimento, o desembaralhador desfaz o embaralhamento realizado pelo embaralhador antes da transmissão e os dados estão prontos para serem processados pelo decodificador EFEC.

Quando o decodificador EFEC está configurado como habilitado, os processos de detecção e correção de erros são realizados e os dados corrigidos encaminhados ao demapeador que, por sua vez, retira a área de dados destinada a FEC do quadro OTU1 e encaminha ao *transceiver* de cliente para que os dados sejam serializados novamente.

6.2 Bloco de Controle

Trata-se de uma interface paralela por meio da qual um processador externo acessa os registros internos do codificador e do decodificador para escrita e leitura. Acessando-se os registros relacionados ao codificador é possível, por exemplo, habilitar e desabilitar a função de codificação; configurar o conteúdo com o qual diversos subcampos do campo SM (*section monitoring*) do quadro OTU serão transmitidos; habilitar e desabilitar o uso do

canal GCC0; habilitar e desabilitar o funcionamento do bloco embaralhador; configurar e monitorar o estado do *transceiver* de cliente; consultar a versão do código gravado no FPGA; etc. Já os registros relacionados ao decodificador permitem habilitar e desabilitar a função de decodificação; ler e zerar o valor de diversos contadores de estatísticas de correção de erro; habilitar, desabilitar e configurar o bloco adicionador de erros; habilitar e desabilitar o funcionamento do bloco desembaralhador; configurar e monitorar o estado do *transceiver* de rede; entre outros exemplos.

Além das funcionalidades citadas, o bloco de controle contém registros relacionados a pedidos de interrupção, aos quais o processador externo recorre quando precisa consultar ou apagar o histórico de pedidos de interrupção ou ainda consultar, habilitar ou desabilitar as máscaras de interrupção.

6.3 Coleta de Dados Para Avaliação de Desempenho

Para viabilizar a avaliação de desempenho dos EFECs, um bloco adicionador de erros foi implementado em paralelo com o fluxo de dados de recepção, entre os blocos desembaralhador e decodificador, como pode ser visto na Figura 6.4. Por meio de acessos de escrita do processador em registros do bloco controle, o adicionador de erros é capaz de gerar taxas de erro de entrada programáveis desde 10^{-3} até $10x10^{-3}$ (ou 10^{-2}), variando em passos de 10^{-3}. Os erros são produzidos invertendo-se bits recebidos, simulando a ocorrência de problemas como ruídos, distorções, atenuações e outros fenômenos que degradam a informação durante os processos de transmissão, propagação pelo canal óptico e recepção. O atenuador óptico do fabricante Hewlett-Packard, mostrado na Figura 6.1, foi inserido no *loop* óptico para aferir o adicionador de erros levando-o a refletir o comportamento do atenuador óptico real.

O procedimento de medição consiste no levantamento de valores de taxa de erro de saída (BER_{out} ou B_{ref}) para uma dada taxa de erro de entrada (BER_{in} ou B_{in}), onde BER_{in} corresponde ao valor medido com o EFEC desligado e BER_{out} corresponde ao valor medido com o EFEC ligado. Não foi possível obter os valores de BER_{in} e BER_{out} por meio de degradação do sinal óptico pelo fato de que o analisador de protocolos Anritsu não mede taxas de erro de entrada acima de $4x10^{-3}$. Dado o elevado desempenho deste EFEC é necessário injetar, de forma controlada e conhecida, taxas de erro de até $10x10^{-3}$

na entrada do decodificador. Como o bloco adicionador de erros permite inserir erros da forma mencionada anteriormente, o mesmo foi utilizado para degradar o sinal recebido.

Configurando-se o bloco adicionador de erros para gerar valores de BER_{in} iguais a $7x10^{-3}$, $8x10^{-3}$, $9x10^{-3}$ e $10x10^{-3}$ obtém-se quatro valores para BER_{out} relacionados, os quais são apresentados na Tabela 6.1. Tais valores foram usados para traçar a curva 3 da Figura 6.5 e são mostrados com um "x" sobre ela. Tanto os pares de valores BER_{in} e BER_{out} em questão quanto a curva obtida a partir dos mesmos são considerados resultados, portanto podem ser encontrados na Seção 6.4.

6.4 Resultados e Discussão

Nesta seção são apresentados tanto os resultados mais relevantes do projeto que deu origem a esta dissertação como também resultados gerados no período de pesquisa e escrita deste documento.

A Figura 6.5 mostra as curvas BER_{in} *versus* BER_{out} para os códigos: RS(255,239) (curva identificada pelo número 1), EFEC I.9 original (curva identificada pelo número 2) e EFEC I.9 modificado (curva identificada pelo número 3). As duas primeiras foram obtidas a partir de [3] e são de referência. A curva do EFEC I.9 modificado foi levantada segundo o procedimento mencionado na Seção 6.3.

De acordo com [33], o ganho de codificação refere-se à diferença na relação sinal-ruído (SNR, *signal-to-noise ratio*) de entrada necessária para se obter uma determinada taxa de erro de saída (BER_{out}), onde a SNR de entrada em questão pode ser expressa, por exemplo, pelo fator Q (*quality factor*). Segundo [34], o valor do fator Q em decibéis pode ser calculado por

$$Q_{dB} = 20\log_{10}(erfc^{-1}(2BER_{out})). \tag{6.1}$$

Ainda segundo [34], o NCG pode ser visto como a diminuição da potência de transmissão necessária para se obter uma dada BER_{out}, graças à codificação, levando-se em consideração o acréscimo de taxa decorrente da transmissão dos dados de FEC. Ao se comparar as equações de cálculo de ganho de codificação (CG) e ganho de codificação líquido (NCG), (6.2) e (6.3) respectivamente, percebe-se que o valor do ganho é penalizado pelo

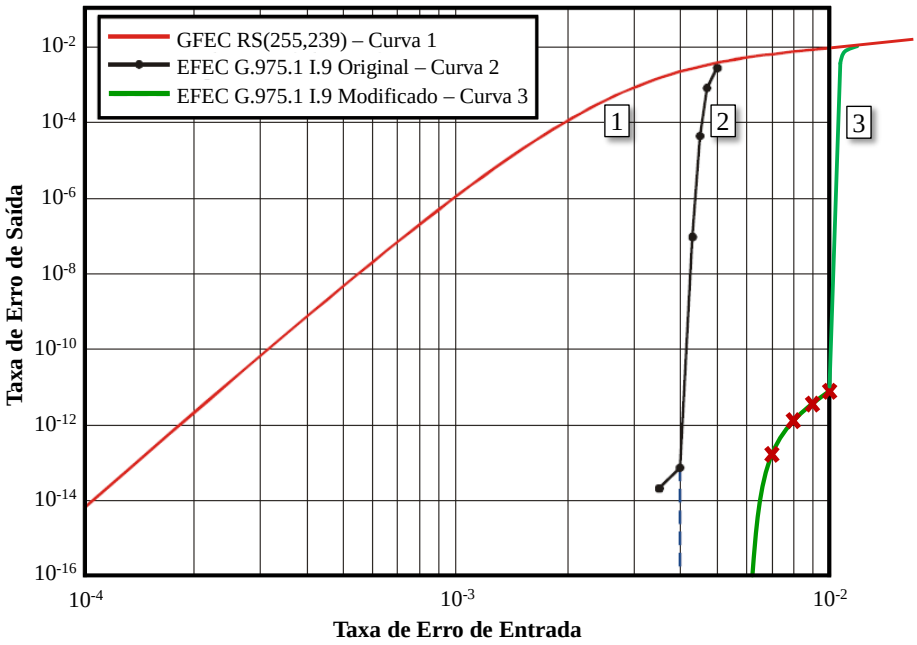

Figura 6.5: Curvas BER_{in} *versus* BER_{out}.

último termo da Equação (6.3), em decorrência do acréscimo de taxa mencionado há pouco.

$$CG_{dB} = 20\log_{10}(erfc^{-1}(2BER_{out})) - (20\log_{10}(erfc^{-1}(2BER_{in}))) \tag{6.2}$$

$$NCG_{dB} = 20\log_{10}(erfc^{-1}(2BER_{out})) - \left(20\log_{10}(erfc^{-1}(2BER_{in}) + 10\log_{10}\left(\frac{1}{R}\right)\right) \tag{6.3}$$

De posse de valores práticos de BER_{in} e BER_{out}, foi possível calcular valores de CG, NCG e fator Q para esse EFEC usando-se as Equações (6.1), (6.2) e (6.3). Os ditos valores são considerados resultados deste trabalho e são apresentados na Tabela 6.1, nos mesmos moldes em que os dados de CG e NCG para o EFEC I.9 original são apresentados pela Tabela I.22 em [3].

Tabela 6.1: Resumo de capacidade de correção do EFEC I.9 modificado.

BER_{in}	BER_{out}	NCG (dB)	CG (dB)	Fator Q (dB)
$7x10^{-3}$	$2x10^{-13}$	8,61	9,41	7,81
$8x10^{-3}$	$2x10^{-12}$	8,39	9,19	7,63
$9x10^{-3}$	$3,5x10^{-12}$	8,45	9,25	7,48
$10x10^{-3}$	$4,7x10^{-12}$	8,54	9,34	7,33

Conforme [35], a evolução nas taxas de transmissão em redes ópticas de acesso está fazendo com que a taxa de erro requerida nesses sistemas mude de $1x10^{-12}$ para $1x10^{-15}$. Posto isso, torna-se conveniente estimar o valor para o NCG para o EFEC I.9 modificado operando a $1x10^{-15}$. Se for considerado que o uso do EFEC em questão leva uma BER_{in} de $6,5x10^{-3}$ à BER_{out} de $1x10^{-15}$ pretendida, o NCG resulta em **9,3 dB**.

Ainda no intuito de contribuir para a caracterização de desempenho do EFEC I.9 modificado, a curva fator Q *versus* BER para esse EFEC foi levantada, tendo como insumo os valores exibidos na Tabela 6.1. Tal curva também é considerada um resultado deste trabalho e é mostrada sob número 4 na Figura 6.6, juntamente com as curvas do EFEC I.9 original (curva 3), do GFEC curva 2) e a curva para o canal não codificado (curva 1).

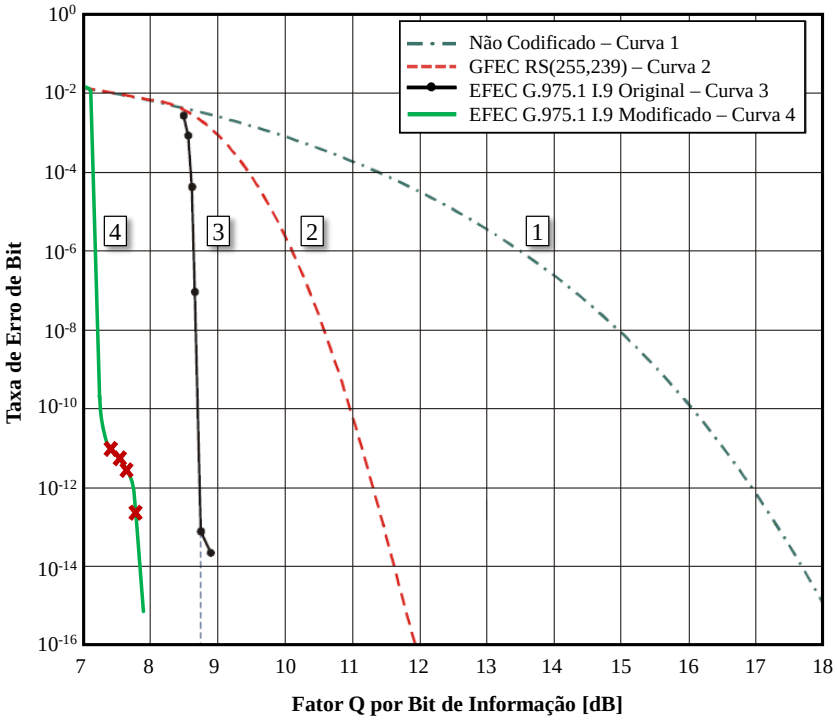

Figura 6.6: Curvas fator Q *versus* BER.

As arquiteturas utilizadas para a implementação prática dos códigos EFEC em FPGA, apresentadas no Capítulo 5, são outro resultado deste trabalho.

Os dados de utilização de recursos lógicos do FPGA também são resultados deste trabalho. Utilizando-se o *software* Quartus, foi feita a síntese individual dos códigos que correspondem ao codificador e o decodificador dos EFECs I.9 original e modificado e os resultados estão expostos nas Tabelas 6.2 e 6.3. O termo ALUTs combinacionais representa a quantidade do recurso tabelas de busca adaptativas (*adaptative look-up tables*) combina-

cionais utilizado, enquanto a sigla BMB corresponde à quantidade do recurso *block memory bits* utilizado.

Tabela 6.2: Ocupação de recursos lógicos - EFEC I.9 original.

Parâmetro		**Codificador**	**Decodificador**
Utilização de Recursos	Registros	10513	6386
	BMB	1251328	1391104
	ALUTs Combinacionais	6224	8042
Desempenho	Frequência Máxima	275,63 MHz	226,96 MHz

Tabela 6.3: Ocupação de recursos lógicos - EFEC I.9 modificado.

Parâmetro		**Codificador**	**Decodificador**
Utilização de Recursos	Registros	13506	15692
	BMB	1482752	1862627
	ALUTs Combinacionais	7021	19493
Desempenho	Frequência Máxima	279,33 MHz	230,36 MHz

O projeto retratado por este trabalho gerou ainda resultados no campo da propriedade intelectual. Os seguintes pedidos de patente foram depositados no Instituto Nacional de Propriedade Industrial (INPI): Arquitetura e Método Para Cálculo dos Vetores q em Códigos EFEC [4], com número de processo BR 10 2014 003615-6, Método de Aceleração para Decodificação Iterativa de Códigos Corretores de Erros e Arquitetura de Decodificador Iterativo [5], com número de processo BR 10 2014 022170-0, Arquitetura de Circuito e Método para Entrelaçamento de *Bits* em Fluxo de Dados Paralelo [6], com número de processo BR 10 2013 026036-3, Arquitetura de Circuito e Método para Processamento Recursivo de Dados Acessados de Memória sem Penalidade de Latência [7], com número de processo BR 10 2014 027625-4, Circuito e Método Gerador de Pulso Síncrono Disparado por Comando Proveniente de Outro Domínio de Relógio [8], com número de processo BR 10 2013 019299-6, Circuito e Método para Armazenamento Síncrono de uma Palavra Digital com Transferência Automática para Outro Domínio de Relógio [9], com número de processo BR 10 2013 031832-9. Além das patentes, também foram depositados junto ao INPI vinte e um pedidos de registro de programa de computador.

Em termos de resultados relacionados a contribuição acadêmica, um artigo intitulado "Ampliação de Ganho de Codificação em um Código EFEC BCH para Comunicações Ópticas" foi apresentado em setembro de 2015 no XXXIII Simpósio Brasileiro de Telecomunicações (SBrT). Esta dissertação em si também pode ser considerada um resultado, uma vez que a mesma reúne as informações do projeto de forma detalhada e organizada.

Por último, mas não menos importante, pode-se citar a geração de conhecimento e capacitação no que se refere a códigos corretores de erros e a implementação dos mesmos em diferentes taxas de transmissão como resultados proporcionados por este trabalho.

Capítulo 7

Conclusões e Projetos Futuros

Baseado nos dados de duas implementações em FPGA de códigos corretores de erro BCH entrelaçados, o presente trabalho mostrou que é possível partir de uma implementação de código BCH para gerar uma segunda implementação com ganho de codificação líquido superior, bastando para tal dividir as palavras-código originais em múltiplas palavras-código (três no caso deste trabalho) e também aumentar a quantidade de informação redundante. Essa estratégia permite que, mesmo que a palavra-código tenha sido dividida, cada parte possa ser processada da mesma forma que uma palavra-código original, o que simplifica o projeto pelo fato de possibilitar a utilização do mesmo circuito base para implementar dois códigos BCH com diferentes capacidades de correção de erro. Apoiado na recomendação G.975.1 do ITU-T [3], as principais características do código EFEC I.9 foram descritas. As arquiteturas desenvolvidas para implementação dos codificadores e decodificadores EFEC I.9 original e modificado foram reveladas, juntamente com o detalhamento de seus sub-blocos e o arranjo experimental usado para testar as referidas implementações.

Este trabalho apresentou ainda um sistema de comunicação clássico acompanhado da explicação de cada um de seus blocos funcionais e fez uma revisão teórica dos conceitos que sustentam a teoria da informação e codificação. Outros assuntos abordados foram as principais causas de degradação dos dados em comunicações ópticas, o protocolo OTN e uma breve perspectiva histórica sobre a evolução dos sistemas de comunicações ópticas até os padrões atuais. Os resultados mais relevantes alcançados pelo projeto que originou esta dissertação, bem como aqueles obtidos durante os processos de pesquisa e escrita da mesma, também foram citados.

Comparando-se a implementação do EFEC I.9 original com a do EFEC I.9 modificado, destaca-se como principal vantagem do segundo código ante o primeiro os acréscimos nos valores de NCG de 0,04 dB para BER_{out} igual a $1x10^{-13}$ e de 0,6 dB para BER_{out} igual a $1x10^{-15}$, o que pode ser considerado expressivo para aplicações de transmissão de longa distância. Em contrapartida, as desvantagens que merecem destaque são o maior consumo de banda (causada pela maior taxa de transmissão) e o maior consumo de recursos do FPGA.

Servindo-se da característica de modularidade da arquitetura, o código EFEC I.9 original também foi implementado em FPGA para operar na taxa de 10,7 Gbps, visando atender o padrão OTU2 de redes OTN. Além disso, está em andamento na Fundação CPqD o projeto de um ASIC (*application-specific integrated circuit*) para tecnologias ópticas de 100 Gbps, no qual o EFEC I.9 original está sendo implementado para operar em OTU4 com aproveitamento da arquitetura mostrada neste trabalho. O circuito integrado em questão se enquadra no estado da arte da microeletrônica e contribui para elevar a tecnologia nacional de equipamentos de comunicações ópticas a um nível ainda mais alto de competitividade.

Como sugestão de projetos futuros, pode-se recomendar a repetição dos experimentos realizados neste trabalho em OTU2, OTU3 ou OTU4, sendo o último mais indicado por se tratar de um padrão mais recente. Outra abordagem poderia ser eleger um outro código diferente do BCH e avaliar os efeitos do aumento de redundância no ganho de codificação.

Dados de [36] indicam um aumento de 20% ao ano no consumo de banda, apontando para a necessidade de evolução na taxa do OTU4. Por conseguinte, trabalhos relacionados a FEC e alinhados com a próxima geração das redes de transporte óptico, possivelmente a uma taxa de 400 Gbps, podem ser uma alternativa interessante de trabalho futuro.

Bibliografia

[1] Cisco, "The Zettabyte Era—Trends and Analysis." Disponível em `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.html`, Acesso em 25 jun. 2015.

[2] ITU-T G.709, "Interfaces for Optical Transport Network (OTN)," February 2012.

[3] ITU-T G.975.1, "Forward Error Correction for High Bit-Rate DWDM Submarine Systems," February 2004.

[4] D. B. d. Carvalho and L. J. R. d. Jesus, "Arquitetura e Método Para Cálculo dos Vetores q em Códigos EFEC," February 2014.

[5] L. J. R. d. Jesus, "Método de Aceleração para Decodificação Iterativa de Códigos Corretores de Erros e Arquitetura de Decodificador Iterativo," September 2014.

[6] E. N. F. Bastos and L. J. R. d. Jesus, "Arquitetura de Circuito e Método para Entrelaçamento de Bits em Fluxo de Dados Paralelo," October 2013.

[7] A. U. d. Lucena, D. B. d. Carvalho, E. N. F. Bastos, G. Ferronato, L. J. R. d. Jesus, M. M. Polidoro, and R. S. Caproni, "Arquitetura de Circuito e Método para Processamento Recursivo de Dados Acessados de Memória sem Penalidade de Latência," November 2014.

[8] A. U. d. Lucena, D. B. d. Carvalho, E. N. F. Bastos, G. Ferronato, L. J. R. d. Jesus, M. M. Polidoro, and R. S. Caproni, "Circuito e Método Gerador de Pulso Síncrono Disparado por Comando Proveniente de Outro Domínio de Relógio," February 2013.

[9] A. U. d. Lucena, D. B. d. Carvalho, E. N. F. Bastos, G. Ferronato, L. J. R. d. Jesus, M. M. Polidoro, and R. S. Caproni, "Circuito e Método para Armazenamento Síncrono de uma Palavra Digital com Transferência Automática para Outro Domínio de Relógio," December 2013.

[10] J. Rochol, *Comunicação de Dados*, vol. 22. 1 ed., 2012.

[11] G. P. Agrawal, *Fiber-Optic Communication Systems*. 4 ed., 2012.

[12] G. Keiser, *Comunicações por Fibras Ópticas*. 4 ed., 2014. Tradução de Márcio Peron Franco de Godoy.

[13] E. Mobilon, "Análise Experimental das Aplicações de Códigos Corretores de Erro em Sistemas de Comunicações Ópticas," masters's thesis, Unicamp, 2002.

[14] R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective*. 2 ed., 2002.

[15] D. van den Borne, *Robust Optical Transmission Systems: Modulation and Equalization*. 1 ed., 2008.

[16] C. A. F. Marques, "Gravação de Redes de Bragg Avançadas em Fibra Óptica," masters's thesis, Universidade de Aveiro, 2008.

[17] J. Evaristo and E. Perdigão, *Introdução à Álgebra Abstrata*. 1 ed., 2002.

[18] J. C. Moreira and P. G. Farrell, *Essentials of Error-Control Coding*. 1 ed., 2006.

[19] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*. 1 ed., 2009.

[20] W. W. Peterson and E. J. Weldon, *Error-Correcting Codes*. 2 ed., 1972.

[21] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. 2 ed., 2004.

[22] M. Borda, *Fundamentals in Information Theory and Coding*. 1 ed., 2011.

[23] O. A. Logachev, A. A. Salnikov, and V. V. Yashchenko, *Boolean Functions in Coding Theory and Cryptography*. 1 ed., 2012.

[24] R. Bernardo, T. R. Tronco, C. L. A. Lessa, and E. Mobilon, "OTN: A Evolução das Redes de Transporte," *EXFO*, 2009.

[25] T. R. Tronco, *Redes de Nova Geração*. 2 ed., 2006.

[26] I. Djordjevic, W. Ryan, and B. Vasic, *Coding for Optical Channels*. 1 ed., 2010.

[27] F. Mayer-Lindenberg, *Dedicated Digital Processors: Methods in Hardware/Software Co-Design*. 1 ed., 2004.

[28] S. Gravano, "Decoding the Triple-Error-Correcting (15,5) Binary BCH Code by the Analytic Solution of the Cubic Error-Locator Polynomial Over GF(2^4)," *International Journal of Electronics*, vol. 68, no. 2, pp. 175–180, 1990.

[29] J. B. Anderson and S. Mohan, *Source and Channel Coding: An Algorithmic Approach*. 1 ed., 2012.

[30] C.-L. Chen, "Formulas for the Solutions of Quadratic Equations Over GF(2^m)," *IEEE Transactions on Information Theory*, vol. IT-28, pp. 792–794, 1982.

[31] A. Katsaros, "Decoding of the (15,7) and (31,21) Binary BCH Codes," *International Journal of Electronics*, vol. 64, no. 4, pp. 637–640, 1988.

[32] ITU-T G.798, "Characteristics of Optical Transport Network Hierarchy Equipment Functional Blocks," December 2012.

[33] K. Kazi, *Optical Networking Standards: A Comprehensive Guide for Professionals*. 1 ed., 2007.

[34] B. P. Smith, *Error-Correcting Codes for Fibre-Optic Communication Systems*. Phd thesis, University of Toronto, 2011.

[35] B. Li, K. J. Larsen, J. J. V. Olmos, D. Zibar, and I. T. Monroy, "Application of Beyond Bound Decoding for High Speed Optical Communications," *ACP/IPOC*, 2013.

[36] Y. Miyamoto, A. Sano, and T. Kobayashi, "The Challenge for the Next Generation OTN Based on 400 Gbps and Beyond," *National Fiber Optic Engineers Conference - Optical Society of America*, 2012.